INTERACTIVE DESIGN OF 3-D PROCESS PLANT LAYOUT WITH AUTOMATIC PIPE ROUTING

by

SUNIL BHALCHANDRA DAMLE

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

SEPTEMBER, 1990

INTERACTIVE DESIGN OF 3-D PROCESS PLANT LAYOUT WITH AUTOMATIC PIPE ROUTING

A thesis submitted

in partial fulfilment of the requirements

for the Degree of

MASTER OF TECHNOLOGY

by

SUNIL BHALCHANDRA DAMLE

to the

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR.

September 1990

ACKNOWLEDGEMENTS

I am extremely grateful to Dr. B. Sahay and Dr. T. Sundarara for their inspiring guidance, constructive criticism and suggesti and for the support and encouragement they have provided during course of this work.

I would like to thank all my friends who made my stay at IIT memorable one. Special thanks are due to Mr. Milind A. Bhandar for all the help he has extended in academic as well as non-acade matters.

Sunil B. Da

CERTIFICATE

This is to certify that the work entitled ' INTERACTIVE DESIGN OF 3-D PROCESS PLANT LAYOUT WITH AUTOMATIC PIPE ROUTING ' has been carried out by Mr. SUNIL BHALCHANDRA DAMLE under our supervision and has not been submitted elsewhere for the award of a degree

Dr. B Sahay,
Professor,
Department of Mechanical Engineering,
Indian Institute of Technology,
Kanpur - 208016, INDIA

Dr. T. Sundararajan,
Assistant Professor,
Department of Mechanical Engineering,
Indian Institute of Technology,
Kanpur - 208016, INDIA

September 1990

ABSTRACT

Process plant design consists of three major steps, viz the process design, strength based design of equipment and pipes, and the layout design of the plant. Each of these is dependant on the others The design of the layout of the process plant is an iterative and time consuming process The time required for this, can be cut short by using a computer aided design system for designing the layout of a process plant.

In the present work, a software is developed, for interactive design of the process plant layout, on the HP-9000/360 series workstation using STARBASE graphics The menu driven, user friendly software has been developed using C programming language in Unix (HP-Ux) environment This software allows the user to develop a model of the plant, modify it as required, and layout the pipes in the plant. The software uses fast interference detection modules to ensure the geometric consistency of the plant model Every addition of a vessel or a pipe to the plant model, and every alteration of an existing component is checked by the system to make sure that it does not interfere with other components of the plant

The software produces three shaded and rendered views of the plant, viz plan, elevation and isometric views, allowing the user to visualise the plant before it is built and foresee any problems that may occur during construction and operation of the plant

The software uses interactive computer graphics for all the above It also has facilities for automatic pipe routing, wherein it lays out a pipe, avoiding all obstacles, without any user intervention.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

A process plant can be defined as a set-up designed and erected to carry out certain chemical engineering processes, mostly to produce chemicals on an industrial scale, typical examples being petrochemicals, fertilisers, etc Apart from the plants producing industrial chemicals, a large number of other installations such as water treatment plants, steam generation plants and drilling fluid circulation systems in oilwell drilling rigs, also belong to the category of process plants Regardless of the chemicals produced or the chemical engineering processes employed such as separation, mixing, decomposition and distillation, each process plant essentially consists of reactors, storage tanks, heat exchangers, pumping equipment and the piping interconnecting these equipment In general, a plant can thus be said to consist of 'vessels' and 'pipes', in addition to the structural elements such as vessel and pipe supports Design and arrangement of these individual vessels and the pipes in the plant area is of prime importance in the context of the overall plant design

With the development of newer processes to produce various chemicals, the complexity of the plant is ever increasing. A modern process plant is not built with the aim of producing a single chemical but the byproducts of the process are typically used as raw materials for producing some other chemicals. This increases the complexity of the plant A typical petrochemicals plant, for instance, consists of a large number of vessels and pipes with a very complex piping system Attempts to save space and reduce the length of the pipes, add to the complexity of the plant This is more so in offshore process plants built on platforms where

space is a prime constraint

As such the process plant layout is an important step towards designing a plant While laying out vessels and piping in a plant due consideration has to be given to the requirements of the process itself, the capacity and the strength design of the pipes, maintainability of the plant, structural design requirements and the usual cost considerations etc It is thus an iterative process with the designer modifying and remodifying the layout to satisfy all the requirements More often than not some of the above requirements may contradict each other For instance, the maintainability and space saving considerations go against each other In such cases the designer therefore has to compromise on some of the requirements Each iteration of the layout process thus produces a better plant layout design However with each change in the layout of the vessels, the affected pipes have to be redesigned and the supporting structures of the vessels have to be reconsidered if the positions of the vessels have been altered

At present the process plant layout is done in the industry at the drawing board using a set of sectional plan views of the plant at various elevations A large set of documentation like process flow sheets, piping isometrics, bill of materials etc is required to be maintained. With every iteration of the layout process, these documents have to be updated Such a method is obviously slow and time consuming, at times forcing the designer to settle for a less than optimum design, for want of time. A strong need exists in the process industry, therefore, for a software that would speed up the whole ~~whole~~ design procedure, by automating it Already some parts of the design process have been automated, for example the automatic thermal stress analysis of the pipes is widely used In ~~the country~~ some softwares are available, at present in the form of object codes, which make use of the geometric modelling techniques to represent a process plant

in the computer's memory and use it mainly for documentation work However, a complete plant design/layout software having abilities to do all the above work of layout of vessels, documentation, stress analysis of the pipes etc as well as automatic routing of pipes, is not developed yet Such a software would enable a designer to design/layout a complete process plant sitting at a terminal and would also be useful in maintenance and operation of the plant. The process of designing and laying out a process plant will be tremendously simplified and speeded up using such a system With the availability of the powerful computer graphic techniques, and suitable workstations, development of such a software is possible

1.1 Problem Definition and Background

An industrial process plant of the most general type is spread over a large area and comprises of several vessels and pipes apart from structural members However while designing the layout of such a plant it is broken up into smaller areas usually on a functional basis depending on the process. Various stages are essentially separated physically from each other and the equipments/vessels required for each stage are clustered in smaller areas called units, from the viewpoint of ease in design, construction and operation In each such unit, the equipment/vessels and the pipes connecting them are laid out in proper sequence The pipes leading to vessels in other such units are laid out upto the exit points on the boundary of the units Once the layout of each unit is complete, the external pipes are routed considering each unit as a box and without bothering about the internal layout of the units Thus, the problem of designing the layout of a process plant reduces to the layout of vessels and pipe routing within a limited area With this consideration, an interactive system on a graphics workstation is intended to be built, which will simplify the process of laying out vessels and

pipes inside a typical unit, with automation of the pipe routing The ultimate aim is the development of a system that will enable the designer to work out the complete layout and analysis of the entire process plant sitting before a terminal A project is underway at CAD centre, I.I.T Kanpur to develop such a system for Engineers India Ltd The complete system as it is projected at present will be able to do the following

(i) Serve as a complete drafting package for interactively laying out vessels and other equipment, and have the ability for manual and completely automated routing of pipes connecting these vessels and equipment It will also enable the incorporation of subsequent modifications to an existing plant

(ii) Maintain and extract data of standard pipes, fittings, and other equipment, these data, in turn, will be used for automatic pipe routing and various other functions

(iii) Display the plant layout as wireframe and surface model with hidden surfaces removed, so as to let the user have a complete feel of the plant before it is built and help him foresee any problems that may arise during construction and operation of the plant

(iv) Communicate with atleast a few analysis packages for pipe design so as to make it a complete design/drafting package for process plant design

(v) Generate useful documents such as bill of materials, piping isometrics etc. to be used in erection work

The system will thus enable the user to design the layout of a process plant once he has the process flow sheets and the overall dimensions of the vessels/equipment and the diameters of the pipes.

1.2 Review of Existing Plant Modelling Systems and Related Topics

Until recently, process plant modelling was attempted with the main aim of automating documentation related to the process plant A large set of documents relating to the plant design, standard equipment used, bill of materials and other documents required for erection and maintenance work such as piping isometrics, material take-off etc, is required to be maintained while designing and constructing a process plant During the iterative process of the design of layout of the plant these documents are required to be updated each time the layout is changed Plant models were developed which contained information about the equipment used and the interrelation between them Small programs were used to extract information from these databases of plants, to produce the required documents Thus, if any change was made to the layout of the plant, only the plant model was needed to be updated

This type of a plant model was not required to contain a detailed geometric model of each component, because the display of the plant as a geometric model was not required The responsibility of ensuring the geometric consistency of the various components in the plant model was left to the designer It was upto him to ensure that the plant is geometrically feasible, and that the various components do not interfere with each other As such, for each component of the plant only a representation understood by the particular software and the user was developed Such plant models served as 'soft copies' of the plant, and had a limited use

With the recent development of powerful computer graphics, it has been possible to build a true geometric model of the plant, and use it to develop a system, which would serve as a drafting package for interactively designing the layout of the plant. The displays showing the various views of the plant, required for such work could be produced from the same geometric model At any stage the

plant layout could be modified and the corresponding changes could be made to the plant model With the availability of the geometric model of the plant, the user has been relieved of the responsibility of ensuring the geometric consistency of the plant Checks to ensure that no two components of the plant interfere with each other, have been built in the system

In 1982, a software based on the above lines was developed at CAD-Centre limited in Cambridge, UK This software, called PDMS (Plant Design and Management System) is being used since then, by Electricite' de France, for its work relating to the layout design of different piping circuits of power stations, which require the use of a large number of pipes Using PDMS it is possible to study the arrangement of pipes to ensure that they do not intersect

While using this system, first large diameter pipes are laid out, and then the smaller diameter pipes are laid out Once the unit is complete, it is difficult to determine from the display only, which, if any, of the pipes interfere The PDMS program uses an automatic interference detection module (CLASHER) which uses the geometric models of the components of the unit to detect interference and provide a list of pipes which interfere and therefore, are not acceptable The PDMS program can also produce piping isometrics which are essential documents for piping erection work However it does not have any facilities for automatic pipe routing The interference detection is done 'off line'

A similar system is also available with Engineers India Limited, which has been marketed by Intergraf Inc This system also derives data from a database of standard equipments and pipe fittings etc used in the process plants Displays of the various views of the plant are possible The system uses a module called ISOGEN which produces piping isometrics It has facilities for automatic dimensioning However in this system too, no facilities for automatic pipe routing

are available.

Automatic routing is a problem which is often encountered in robot motion planning, and in routing of ICs (Integrated Circuits). In motion planning for robots, a robot arm is required to move in 3-D space without colliding with the obstacles However, in this case, more stress is laid on the kinetics and dynamics of the robot arm related to the path. In the case of mobile robots, the robots are considered to be small objects as compared to the obstacles For this, proportional shrinking of the robot and enlargement of the obstacles is carried out, and then a path for a small object, ideally a point, is searched, avoiding obstacles This problem is mostly solved as a 2-D problem, the point representing the robot, moving in a plane There is no restriction on the turning angle, where the path of the robot is required to take a turn

In the design of layout of integrated circuits the signal nets have to be routed for intra-block and inter-block communication. Routing area is a major proportion of the total area of an integrated circuit Similar is the case with the piping in a process plant In the layout design for the ICs, it is attempted to minimise the overall area of the IC by minimising the routing area. The blocks to be connected by a network of routes, are moved, and positioned at suitable locations, to achieve this [1,2,3] However, in case of pipe layout, the positions and orientations of the vessels to be connected by the pipes are more or less fixed, before the pipes are routed Also routing in the ICs is essentially a 2-D problem, and ideally a single line of a signal net route has negligible width.

The algorithms used for the above problems, can not be used for pipe routing, because the pipe route, to be feasible, must allow sufficient 'turning radius' for the pipe bend, and there is a restriction that the pipe route can not change direction at any arbitrary angle These restrictions are not applicable in

case of the two problems discussed above

In the present work, Dijkstra's Algorithm for shortest path has been implemented for automatic routing of pipes [4] It is described in detail under automatic routing In addition to the automatic routing algorithm, for the graphical representation of vessels, straight pipe pieces, and pipe bends, in various views, certain standard matrix transformations are used Also, for achieving a realistic display the STRIP Z-buffer algorithm is employed The relevant matrix transformations and the application of the Z-buffer algorithm are described in [5,6,7]

1.3 Scope of Study , Objectives

The aim of the present thesis was to develop a system that would be a first step towards a more versatile system having capabilities as explained above The software developed as a part of this study is expected to perform the following tasks

(i) Serve as a drafting package for interactively specifying plant dimensions, positioning and orientation of vessels, addition and deletion of vessels, deletion of the pipes etc

(ii) Assist the user in manual routing of pipes in the plant by interactively picking points in 3-D space through a display of the plant. Provide the facility of automatic routing of the pipes with minimisation of number of bends.

(iii) Graphic display for all the above interactive functions and display of the whole plant in various views

(iv) General file management functions for the system through a user friendly menu, for loading and saving the plant files, opening new plant files etc

1.3.1 Assumptions and Limitations

The following assumptions were made while developing the system

(i) The vessels/equipment are only cylindrical In an actual process plant most of the vessels and other equipment such as reactors, columns, heat exchangers, pump and blower housings, motor housings etc are indeed cylindrical, and can therefore, be modelled as cylinders, in the plant model Equipment such as valves etc can be modelled as a combination of cylinders Some vessels can be modelled as spheres. Such vessels are not considered in the present system and can be added later on without much difficulty

(ii) Pipe routing in a process plant is done usually through piperacks which are placed suitably between rows of vessels Though the concept of piperack is included in the system, it is not used in the autorouting function. The extension of autorouting module, taking into account the piperacks is rather straightforward.

(iii) There are no structural members in the plant. Structural members such as vessel and pipe supports, pipe clamps etc. are not considered in the plant model For this a detailed plant description and modelling will be necessary which is not of prime importance in the present work. The plant is assumed to be a cube within which all the vessels and pipes are laid out. This is done for the ease of display without loss of clarity, and it does not affect the design of the system in any way.

(iv) The pipes are routed using 90° pipe bends only. This assumption is very practical because in actual process plants pipe inlets to and outlets from the vessels are in most cases in principal directions and the pipes are routed using standard 90° pipe bends only Only in some cases the vessels to be connected by a

pipe are situated far away from each other and in such cases considerable saving can be achieved by using bends which do not have 90° turning angle In the present work, however, the plant is considered to be small and the case of far off vessels can be dealt with by considering the vessels to be situated in two different units of the plant, routing the pipes in each of the units separately and then routing the external pipe between the two exit points of the two units Such an arrangement using 90° pipe bends only, improves the maintainability of the plant considerably

1.4 General Outline of the System

The system is implemented using C programming language in Unix (hp-ux) environment on HP 9000/360 series workstation, using starbase graphics It is a menu driven system with main menu options of File, Modify, Display and Route, for opening/loading/saving of the plant files, modifying existing plant model. displaying various views of the plant, and manual/automatic routing of the pipes The system is fully interactive and at every stage during opening new plant files, modifying existing ones, routing etc. it provides several options to the user, and takes a suitable course of action based on the option selected For all the functions the user would be able to see the effect of the option he has chosen in the display. The user also has the option to negate any changes made by him either by asking the system not to enter the changes made to the plant model, or by not saving the changes made during the current work session. Thus the user has a complete control over the plant model through the system.

CHAPTER 2

DEVELOPMENT OF THE PLANT MODELLING SYSTEM

2.1 Requirements of the Plant Modelling System

For the system to be able to perform as explained previously it must have the following facilities

(i) Representation of the plant in the computer's memory This is a vital step and the whole implementation of the system depends on it.

(ii) Conversion of the represented plant and its components into wireframe and surface models suitable for display, interference detection and routing functions

(iii) Obtaining projections of these wireframe and surface models on suitable planes to let the user have various views of the plant with hidden surfaces removed

(iv) Assisting the user to interact graphically for prescribing input regarding placement of vessels etc and manipulating the plant representation in response to the changes specified by the user in respect of addition/deletion/movement of vessels etc

(v) Interactively picking up user specified points in 3-D space from the screen, during manual routing, checking the feasibility of the pipe through interference detection, building the pipe model and including it in the plant model.

(vi) Interactively picking up a pair of user specified points in 3-D space and

routing a pipe of given diameter on a fully automatic basis avoiding obstacles and with due consideration to the direction of the pipe at the two points, specified by the user The pipe route generated by the system must require minimum number of pipe bends

(vii) For all the above, the system needs facilities for interference detection, i e given a vessel it must be able to detect whether sufficient surface to surface clearance is available between this component and all the other equipment and pipes in the plant For a newly routed pipe, it must be able to check whether sufficient surface to surface clearance is available between the pipe and all the components of the plant The system must be able to do this very fast, because as explained later in chapter 4, under interference detection and autorouting a very large number of interference checks have to be carried out during all the above functions, especially the autorouting

2.2 Overview of the Plant Modelling System Menu Options

The menu driven interactive system developed in the present work satisfies the above mentioned requirements and assists the user in building a plant model though a main menu The options contained in the main menu are :

File

Modify

Display

Route

Each of the above options have submenus which provide the user a complete control over defining a plant, modifying it, obtaining displays and for automatic and manual routing of pipes Use and function of each option is described below.

(i) File :- Choice of this command from the main menu leads to the following submenu options

* Load - With this option a plant file from the disc can be loaded into the core memory There is an option to the user to save or discard the currently loaded file All the data of the plant regarding the vessels, pipes, piperack, plant room dimensions etc which are stored in a compact form in the plant file, are read into suitable structures by the system

* Save - With this option, the currently loaded plant file in the memory is written to the disc file of the same plant Invoking this option is a must for saving the changes made in the plant model during the current work session

* New :- With this option, the user can open a new plant file in the memory The overall dimensions of the room and the piperack data can be interactively entered to the model The newly opened file has to be named by the user and saved on to disc using the save option, failing which the file is discarded

(ii) Modify - By selecting this command, the user can interactively layout the vessels in a plant. The submenu options available under this command are .

* Add vessels : Using this option the user can interactively add vessels to an existing plant This can be done using two options provided by the system The user can

a) directly specify the diameter and the end points of the centre line of the cylindrical vessel, or

b) specify the diameter and length of the vessel A vessel is shown at a specific location on the display and its location reads out at the bottom of the screen The user can then move it with graphic interaction to any place in the plant. Every time the vessel is moved it is shown at the new position in plan and

elevation views and the new coordinates are shown at the bottom of the screen.

In both the cases, the system checks the freshely added vessel for interference with any of the earlier placed vessels or pipes in the plant, the piperack and the walls of the plant User has the option to add the newly displayed vessel to the plant or discard it

* Move vessel :- Using this option the user can interactively move any vessel in the plant Again it is possible to add the vessel at new position to the plant model, discarding the vessel at old position or leave it at the old position If the user chooses the new position or new orientation of the vessel, the pipes connected to the vessel are discarded and have to be routed again for new position

In this case too, plan and elevation views of the plant are shown side by side with the display of the moving vessel, the current coordinates of the vessel being shown at the bottom of the screen, numerically

* Delete vessels :- In this option, two views of the plant with pipe and vessel numbers are displayed and a selected vessel can be deleted from plant model The pipes connected to that vessel are automatically deleted

* Delete pipes :- The display is same as above. A selected pipe can be deleted from the plant model

(iii) Display - This option is used to obtain various views of the plant. Three possible views are

* Isometric view

* Plan view

* Elevation view

In each view, the plant model can be displayed as wireframe or a shaded surface model with hidden surfaces removed. It can be specified by the user with the pull down menu

(iv) Routing - This option is used to route pipes in the plant. The submenu options under this heading are

* Manual Routing - In this option a pipe can be routed in the plant by interactively specifying points in 3-D space which serve as guide points for the pipe layout The points are entered using a pair of cursors in the two views of the plant displayed on the screen side by side The user can opt for a wireframe or surface model display. The cursor can be moved using cursor control keys and using the ENTER key a series of the points can be entered When all the points are entered the system checks whether a pipe can be laid out as specified and also checks for the interference The user has the option to build and add pipe to the plant model or discard it

If it is not possible for the system to build the pipe or if it interferes with any equipment/pipe in the plant it reports so and discards the array of points entered

* Autorouting :- Using this option the user can specify the starting and the end point of an intended pipe and ask the system to route the pipe on a fully automatic basis The system then routes the pipe avoiding all the obstacles in the plant while maintaining a predetermined clearance. The autorouting is based on the consideration of minimum number of bends between the starting point and the destination point. If the system is unable to find a path for routing the pipe it reports so.

2.3 Steps Involved in the Development of the Plant Modelling System

The important components of the plant modelling software are

i Represention of the plant

ii Display

iii Interference detection

iv. Manual routing of pipes.

v Automatic routing of pipes

These functions are indeed interdependent, and the requirements of each are dictated by the relations they bear with the other functions and also the overall requirements of the plant modelling system The details of each function, its interrelation with the rest of the system, requirements and performance expected and alternative solutions available for completing the same task with relative merits and demerits are described in the ensuing chapters. The features relating to the plant data representation and the display are discussed in chapter 3 Functions concerning the pipe routing, namely, the manual and automatic pipe routing functions and the interference detection procedures are described in chapter 4 The application of the plant modelling software and the illustration of a few case studies are presented in chapter 5

CHAPTER 3

PLANT REPRESENTATION AND DISPLAY FEATURES

3.1 Representation of the Plant

A complete description of the plant must contain represention of the plant room, vessels, pipes, piperack space and the relationship between these such as the connection between the pipes and vessels, occupation of piperack space by pipes etc. It must have the following characteristics :

i) It must be memory efficient Since a typical plant contains a large number of vessels and pipes, this is an important requirement

ii) It must be suitable for efficient conversion to the corresponding surface model of the plant for display and other related functions Since display of the plant is required in all the functions used for manipulation of the plant, as well as routing, obtaining a quick display of the surface model of the plant is of prime importance for overall performance of the system.

iii) It must be directly usable by the interference detection function because a very large number of interference checks have to be made in a very short time. Needless to say, too much time spent in data conversion is not affordable in this problem.

iv) It must be suitable for easy modification of the plant model and for saving as a disc file and reloading from the disc file.

A possible way to store the various components of the plant is to store

them as surface models This would help in producing a quick display of the plant model However, the storage space needed for this is very large even for a single vessel With a large number of vessels and pipes in the plant, such data storage becomes very expensive Hence, the following representation for the plant components has been adopted, though it requires conversion of stored data to surface model for display and thereby slows down the display process

The system uses a C structure named *part*, which is defined as

```
struct part {
        struct point pt1,
        struct point pt2,
        int          idir,
        int          odir,
        float        dia,
        struct part  *next,
       },
```

where structure *point* contains the three coordinates of a point and is defined as

```
struct  point {
        float  x,
        float  y,
        float  z;
       };
```

These two structures are the basic building blocks in the representation of pipes and vessels as described below

* Vessels - Vessels in the plant are modelled as cylinder. A cylinder can be efficiently stored in the structure *part* described above In the structures *pt1* and *pt2*, the coordinates of the endpoints of the axis segment of the cylinder are stored The diameter of the cylinder is stored in *dia* Direction of the axis of the cylinder is stored in *idir* and *odir* For cylinders oriented in principal directions, the relevant directions are stored as 1 or -1 for x-direction, 2 or -2 for y-direction, 3 or -3 for z-direction and as 7 for generally oriented axis The pointer *next* is kept free Thus a cylinder in space is completely defined by this structure The surface of the cylinder can be generated as and when required by display routines All the vessels in the plant are stored in an array of these structures

* Pipes - Pipes can be considered as a series of straight pipe pieces and pipe bends Radius of the bend is taken as 1 5 times the diameter of the pipe Straight pipe pieces are stored as cylinders as mentioned earlier A pipe bend is modelled as a quarter of a torus It can also be stored in the same structure *part* The diameter is stored in *dia*, endpoints are stored in *pt1* and *pt2*, and the directions of the tangents to the axis arc at the two endpoints are stored in *idir* and *odir* The elements *idir*, *odir*, *pt1*, *pt2* and *dia* completely define a bend as shown in figure 3.1 The surface of the bend can be generated for display from this data

The pipe itself is represented as a linear linked list of these structures, each structure representing a straight or a bent piece Pointer to *part* structure, *next* which points to the structure representing the next connected piece of the pipe is used for this The endpoints of the axis segment of the pipe piece are not interchangeable because a convention is followed that the endpoint represented by *pt2* of a piece is connected to the endpoint represented by *pt1* of the next

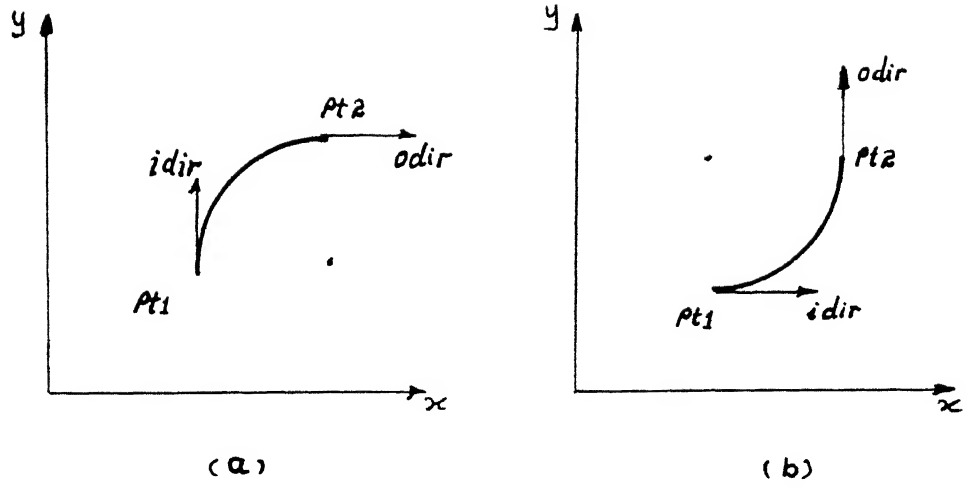


Figure 3.1 Representation of the Pipe Bend.

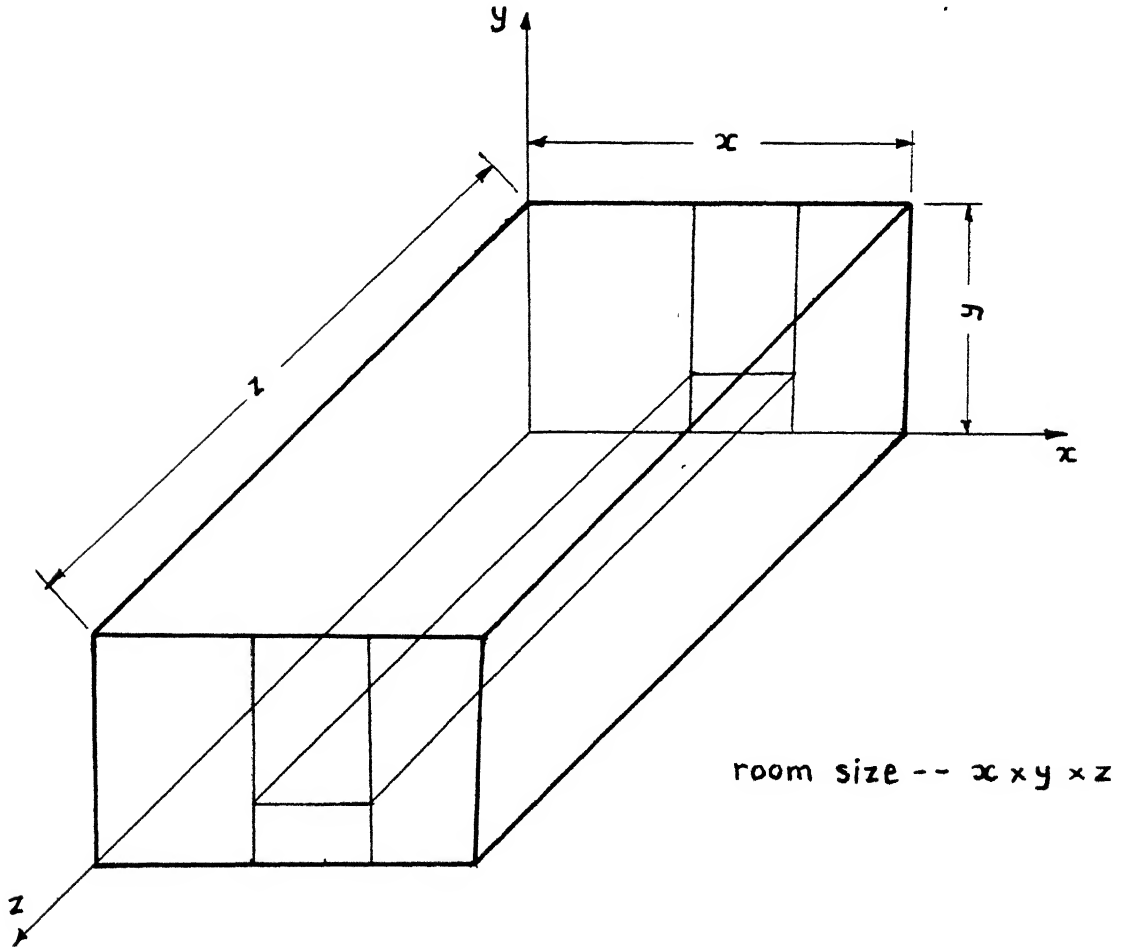


Figure 3.2 Coordinate System for the plant unit.

structure of the linked list All the pipes in the plant are stored as an array of pointers to the first structures of the linked lists representing the pipes

* Piperack - Structural members are not included in the plant model Hence piperack is represented simply as an area where no vessels are allowed. It is stored as its width along x-direction, x-coordinate of the center line of piperack space and elevation from the ground

* Vessel and Pipe Connections :- Each vessel in the plant may have some pipes connected to it This association of pipes and vessels is stored in a 2-D array of pipe serial numbers, the number of rows of the array being equal to the number of vessels in the plant Each row contains the pipe serial numbers of all the pipes connected to that particular vessel number.

* Coordinate System :- A right handed Cartesian coordinate system with origin at one of the bottom corners of the plant room is used for the plant model (figure 3.2) The piperack axis is parallel to the z-axis and the y-axis is along the vertical direction. The plant room itself is stored as vertices of the cube representing the room.

Thus only one structure is used to store all the vessels and pipes in the plant This representation is highly suitable for interference detection calculations since all of its elements, *pt1, pt2, idir, odir* and *dia* are directly usable in these routines as explained in the subsection on interference detection.

This representation consumes very little storage space because each vessel or piece of pipe, straight or bent, is stored in 40 bytes only (3*4 +3*4+4+4+4+4+4) While storing a plant model as a disc file if the number of pipes in the plant is very large, considerable saving in space can be obtained by not writing to disc the point structure representing the second point (pt2) of the structures *part*

corresponding to the pipe pieces This can be done because the first point (*pt1*) of the next piece of the pipe is the same as *pt2* of the earlier pipe piece as explained before Thus for each pipe piece a saving of 12 bytes is effected

As explained in the subsection on Display, the surface model of the vessel or pipe can be generated using this representation For generating a surface model of a pipe, a surface model of each piece of the pipe is generated sequentially Data conversion from this representation to surface model representation essentially involves generation of surface patches that make up the surface of the vessel or pipe. Each patch is represented as a plane polygon in space and is temporarily stored as the coordinates of its vertices before being used in display functions Generation of these surface patches is described in detail in the section on Display

In case of manual routing, the user enters a series of guide points for the pipe to be routed through While building the pipe based on these guide points, the system guides the axis of the pipe and generates the endpoints of the axis segment of each piece and stores them into the respective structures. It can readily be seen that generation of elements *idir* and *odir* for each piece, from the series of guide points is very easy Similarly, in autorouting function the system generates by itself a series of guide points Thus, the representation is most natural for building a pipe model in manual and autorouting functions It also satsisfies all the requirements stated at the beginning of the subsection and is best suited for the present work

3.2 Display

This section discusses the display features of the system in relation to the data conversion of plant model for display, i e generation of surface and

wireframe models of the plant from the representation used by the sustem, projection of the generated model on various planes for display, Z-buffer algorithm for hidden surface removal to achieve certain amount of visual realism in display etc

For the purpose of display, the vessel and the pipes in the plant are assumed to be filled solids, and only external surfaces are generated while obtaining display The internal surfaces of the actual hollow vessels and pipes are neglected because the pipes and vessels are made of opaque material and hence the representation by only external surface is adequate for display purposes Two types of geometric models are used while displaying the plant

Firstly, a wireframe model is created, in which the surface of each component of the plant is broken up into a number of polygonal surface patches and each surface patch is stored as a group of edges binding it. If so desired by the user, this wireframe model is subsequently extended to a surface model, which is particularly suitable for shaded and rendered displays with hidden surfaces removed

Generation of both models from the representation of the plant is discussed below

3.2.1 Data Conversion to Generate Wireframe and Surface Models

As explained earlier the vessel and pipes in the plant are stored using the c-structre part For displaying any view of the plant either as wireframe or surface model. it is necessary to generate the surfaces of all vessels and all pieces of each pipe in the plant and then obtain projections of these surfaces on the required planes

Two types of surfaces are required to be generated viz, the surface of a vessel or straight pipe piece modelled as a cylinder and that of a pipe bend modelled as a quarter torus In each case, a set of plane polygonal patches which make up a surface closely approximating the actual surface are generated as described below In general, for any given curved surface, more the number of the patches (i e smaller the patches), better is the approximation of the actual surface However, a large number of patches require more processing time for display and hence a compromise has to be made to obtain a reasonably accurate approximation fast

Consider a line segment p_1p_2 where p_1 is $p_1(x_1,y_1,z_1)$ and p_2 is $p_2(x_2,y_2,z_2)$, generally oriented/positioned in space (figure 3.3) By using some transformations it can be aligned with the y-axis, having one endpoint say p_1 at the origin. This can be achieved by applying a set of transformations sequencially, as described below using homogeneous coordinates

(i) Translation of the point P_1 to the origin which in homogeneous coordinates can be obtained by the transformation matrix .

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -dx & -dy & -dz & 1 \end{bmatrix} \quad \text{---- (3 1)}$$

This transforms P_1P_2 to $P_1'P_2'$. (figure 3.3(a))

(ii) Rotation about y-axis by an angle α to bring segment $P_1'P_2'$ in the z-plane by the transformation matrix

$$R_y = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{---- (3.2)}$$

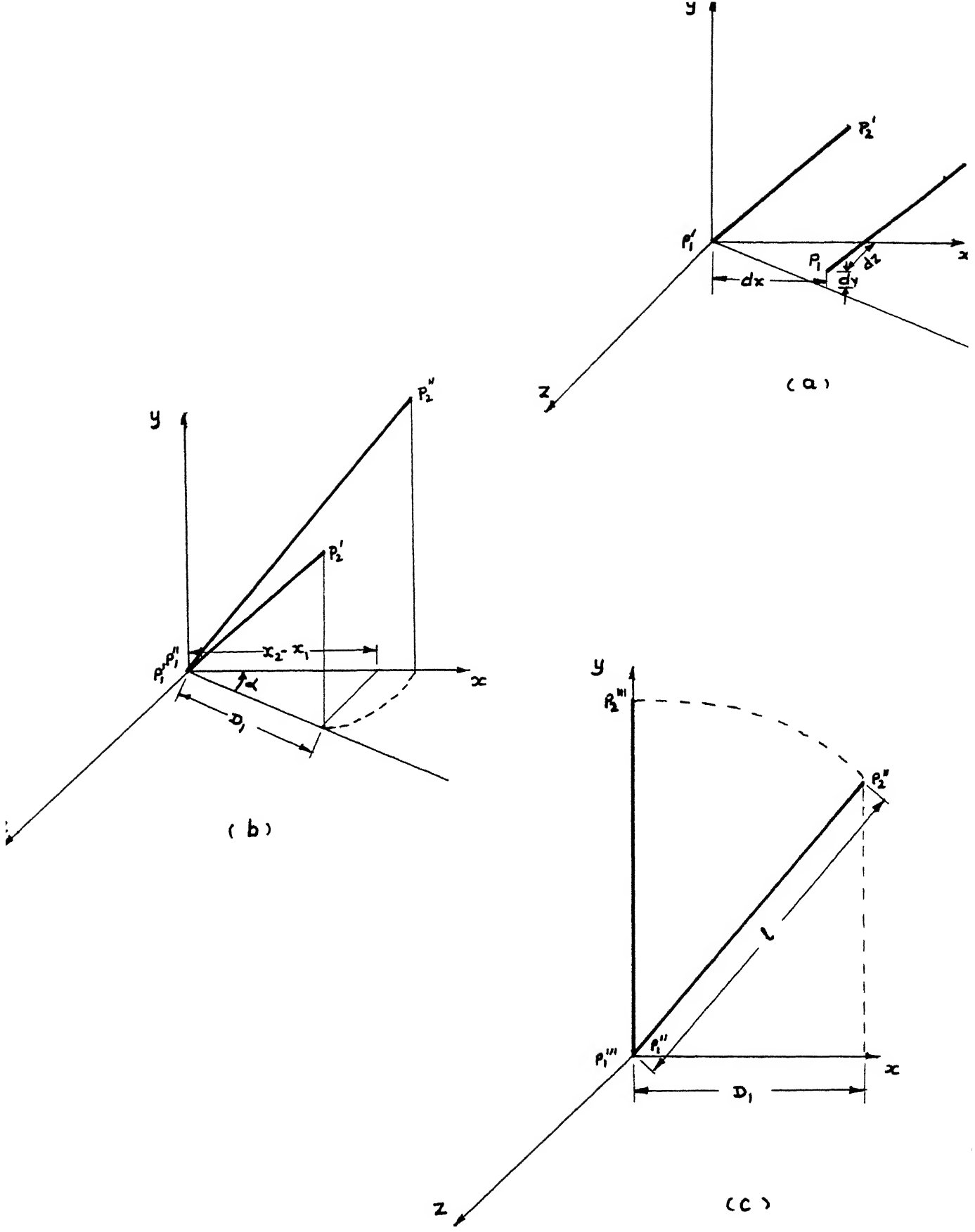


Figure 3.3 Transformations of a line segment.

where the angle α is obtained as

$$\sin \alpha = \frac{z_2 - z_1}{D_1} \quad \text{where } D_1 = \left((z_2 - z_1)^2 + (x_2 - x_1)^2 \right)^{\frac{1}{2}} \text{ ---- (3.3)}$$

This transforms $P_1'P_2'$ to $P_1''P_2''$ where $P_2'' = P_2''(D_1, y_2-y_1, 0)$ (figure 3.3(b))

(iii) Rotation about z-axis by an angle β to align segment $P_1''P_2''$ with y-axis by the transformation matrix

$$R_z = \begin{bmatrix} \cos\beta & \sin\beta & 0 & 0 \\ -\sin\beta & \cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{---- (3.4)}$$

where the angle β is obtained from the relation

$$\sin \beta = \frac{D_1}{l} \quad \text{where } l = \left((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 \right)^{\frac{1}{2}} \text{ ---- (3.5)}$$

This transforms $P_1''P_2''$ to $P_1'''P_2'''$. (figure 3.3(c))

By composition of transformations, all the three transformation matrices T, R_y and R_z can be combined together into a single composite transformation matrix S which transforms segment P_1P_2 to $P_1'''P_2'''$, given by the expression

$$S = T\, R_y\, R_z \quad \text{---- (3.6)}$$

All the elements of the matrices T, R_y, R_z and hence those of the matrix S can be obtained from the coordinates of the points P_1 and P_2. This transformation can be looked upon as a change of coordinate system in which the line segment P_1P_2 lies along the y-axis with point P_1 at the origin.

The aforementioned transformation of a line segment forms the basis of the transformation required to generate the surfaces of straight pipe pieces and pipe bends. The basic idea is to position one endpoint of the axis of the pipe piece represented in element *pt1* of the structre *part* corresponding to the concerned

component, at the origin and rotate the component to an orientation suitable for easy generation of points on the surface of the component The generated points are then ordered appropriately and are used to generate edges for wireframe model or polygonal patches for surface model

3.2.2 Generation of the Surface of a Vessel or a Straight Pipe Piece

The cylinder representing a vessel or a straight pipe piece is stored in a structure part The coordinates of the end points are available in *pt1* and *pt2* The axis segment of the cylinder is transformed as discussed above in case of line segment P_1P_2, by a matrix S The elements of S are computed using the coordinates of the endpoints of axis segment After transformation, the cylinder is thus positioned as shown in figure 3 4 For this cylinder, it is now easy to generate coordinates of points on the surface Thirty six points on the circumference of the flat end faces are generated, and the corresponding pairs of points on each circumference are used to generate a rectangular patch, e g patch number 1 is 1-2-2'-1', patch number 2 is 2-3-3'-2' and so on, all around the curved surface of the cylinder (figure 3 5) Each of the flat end faces is represented by a polygonal patch of thirty six vertices. These. together with the patches for the curved surface make up the complete surface of the cylinder Each polygonal patch is stored as an ordered sequence of vertices

Applying an inverse transformation to this cylinder gives the vessel in the original coordinate system. Hence the surface patches of the original cylinder can be obtained by applying inverse transformation using matrix S^{-1} to all the surface patches generated.

$$S^{-1} = R_z^{-1} R_y^{-1} T^{-1} \quad \text{---- (3 7)}$$

Each surface patch is transformed by applying transformation matrix S^{-1} to all the

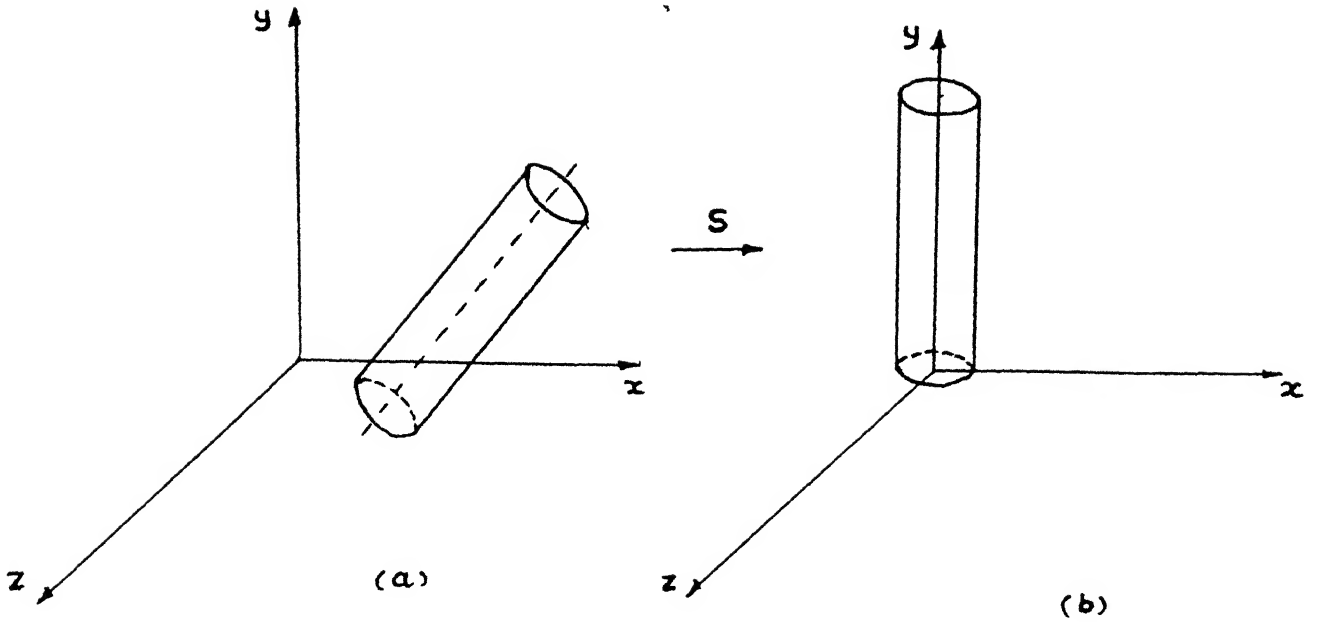


Figure 3.4 Transformation of a cylinder

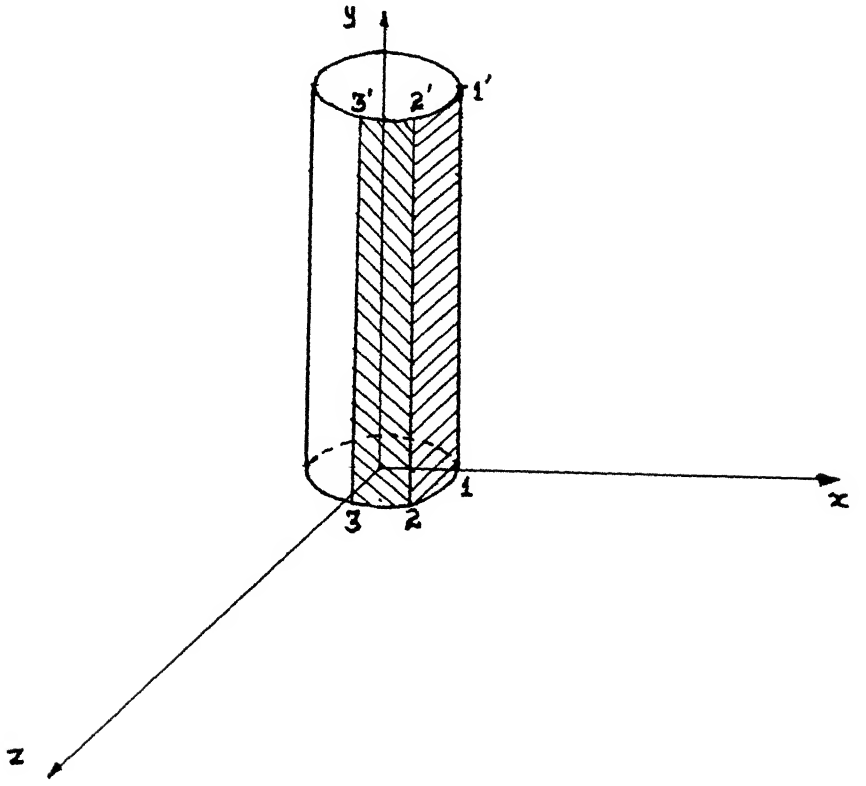


Figure 3.5 Surface model of a cylinder

vertices of that patch

The rotational transformation matrices R_y and R_z are orthogonal matrices. Hence their inverses can be obtained by simply transposing them Inverse of the translation matrix T is

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix} \quad \text{---- (3 8)}$$

Thus, a time consuming matrix inversion process for the matrix S is avoided At the end of this inverse transformation process, the surface patches approximating the surfaces of a cylinder are ready for projection transformations discussed below

For generating a wireframe model of a cylinder, all the above steps except the generation of polygonal patches by pairing of vertices are carried out. Instead of generating surface patches, the points on the circumference of the flat end faces are used to obtain a wireframe model The two end faces are stored as hollow polygons of 36 points and the curved surface is represented by lines connecting pairs of selected points from the two end circles e g 1-1', 2-2' etc. Rest of the process is the same as for the surface model, including the inverse transformation by matrix S^{-1}

3.2.3 Generation of the Surface of a Pipe Bend

The pipe bend is modelled as a quarter torus. The coordinates of the end points of the axis (arc of a circle of radius = 1.5 * diameter of the pipe) are available in *pt1* and *pt2* and the directions of tangents to the arc at the two end points are available in *idir* and *odir*. As the pipes are routed along principle directions using only 90° bends, the axis arcs of all the pipe bends are in planes parallel to the principal planes.

For generating the surface patches, the procedure adopted is similar to that used for generating the surface patches of a cylinder, i.e, transforming the bends to a suitable coordinate system in which the surface points and the patches can be generated easily and then transforming these patches to the original coordinate system using the inverse transformation For this, the bend is characterised by two line segments AB and AC as shown in figure 3.6(a). Point A corresponds to *pt1* and point C corresponds to *pt2* of the structure *part* representing the bend The pair of segments AB and AC are transformed such that segment AB is along Y-axis or X-axis depending on the direction of tangents indicated in *idir* and *odir* elements of the structure *part*; e g if *idir* =1 i e +X direction, then segment AB is aligned with +X- axis and if *idir* = 2 segment AB is aligned with +Y- axis This is done for ease of calculation of the transformation matrices The point A coincides with origin and segment AC is in the Z-plane in the first quadrant as shown in figure 3.6(b),(c)

After such a transformation, the axis arc of the pipe bend can be in two positions as shown in figure 3.6(b),(c), depending on whether the segment AB is aligned with X-axis or Y-axis For this transformation, a matrix S_1 (which is similar to S discussed above) is first used to align AB along the required axis If segment AB is to be aligned with Y-axis

$$S_1 = T \; R_y \; R_z \quad \text{---- (3 9)}$$

and if AB is to be aligned with X-axis

$$S_1 = T \; R_x \; R_z \quad \text{---- (3.10)}$$

where R_x and R_y represent the matrices for rotation about the X-axis and Y-axis respectively As the segment AB will always be parallel to one of the principal directions, the calculation of the elements of the rotation matrices R_x ,

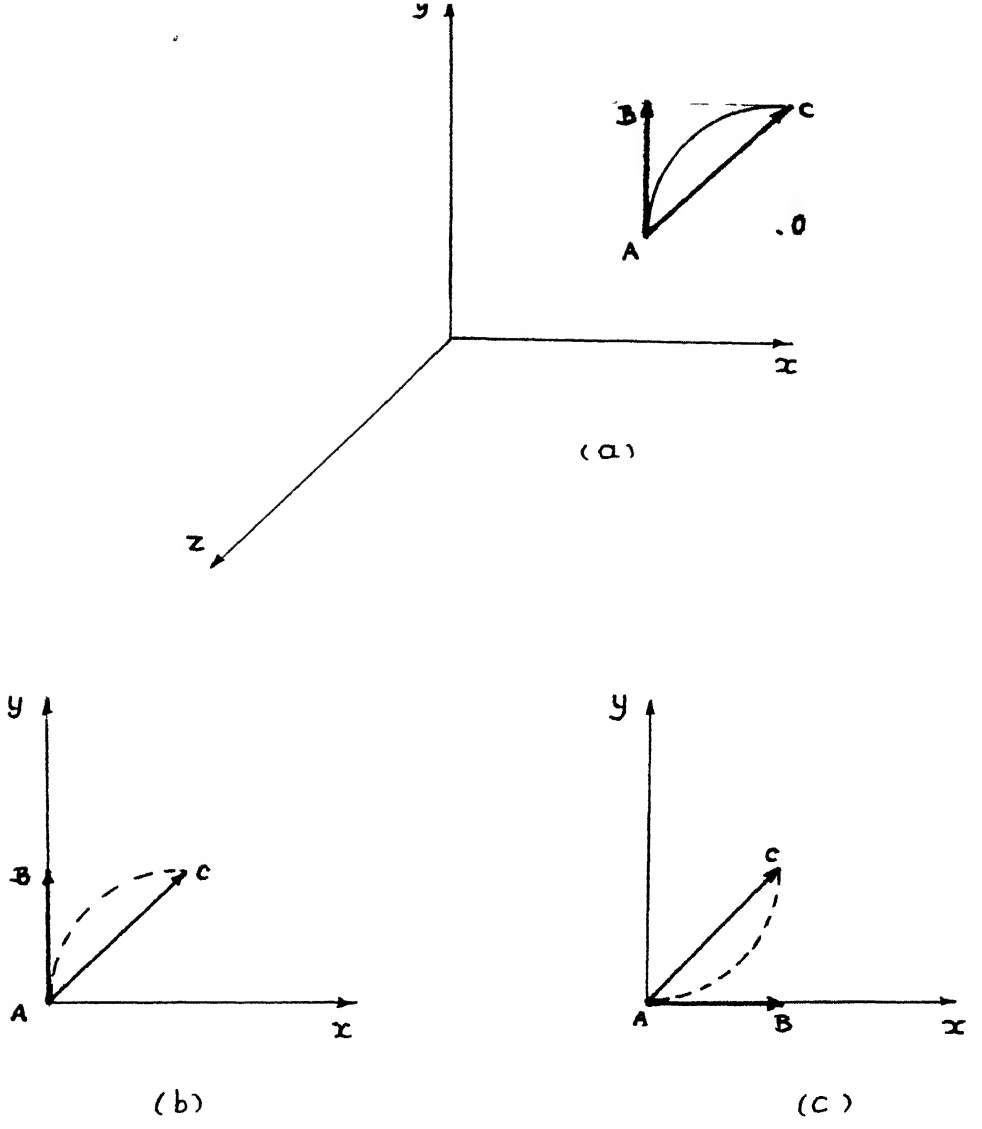


Figure 3.6 Transformation of a pipe bend.

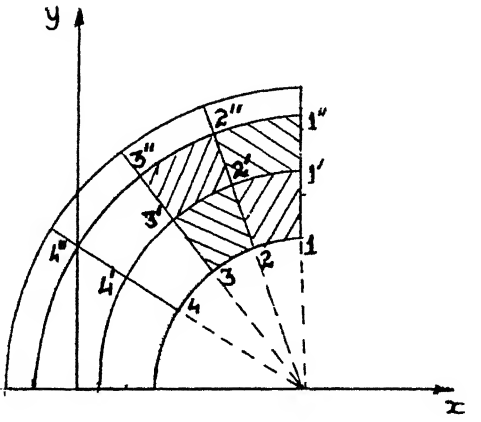


Figure 3.7 Surface Model of a pipe bend.

R_y and R_z is far easier than that for a cylinder generally oriented in space All the elements of these matrices will always be either 0 or -1 or +1, since the angle of rotation α and β will always be 0 or multiples of 90°.

Further, rotation transformation about the axis along which the segment AB has been oriented, is needed to bring point C in the desired position Transformation matrix denoted by R_{y_2} or R_{x_2} is used for this rotation, and this matrix also has elements which are 0, -1 or 1 for reasons explained above Thus the computation of the overall transformation matrix in this case is faster than that for a straight pipe piece. The overall transformation matrix S_2 is

$$S_2 = S_1 R_{y_2} \quad \text{or} \quad S_2 = S_1 R_{x_2} \quad \text{---- (3 11)}$$

Alernatively, $S_2 = T R_y R_z R_{y_2}$ or $S_2 = T R_x R_z R_{x_2}$ ---- (3.12)

In the transformed coordinate system the points on the surface of the bend are generated Thirty six points on each plane end circle are generated and each plane surface is represented by a polygonal patch of 36 vertices Plane quadrilateral patches which make up the approximation for the doubly curved surface of the bend are generated by breaking up the surface in strips parallel to the axis arc of the bend and then breaking up each strip into quadrilateral patches For this, appropriate groups of four points on the surface (figure 3.7) are used as vertices of the quadrilateral patches, e.g 1-2-2'-1', 2-3-3'-2' ..and 1'-2'-2''-1'', 2'-3'-3''-2'' . . .and so on.

These quadrilateral patches and the polygonal patches representing the end faces, together make up the surface model of the pipe bend. The vertices of the pipe bend are then transformed back to the original coordinate system with inverse transformation matrix S_2^{-1} .

where $S_Z^{-1} = R_{y_Z}^{-1} R_z^{-1} R_y^{-1} T^{-1}$ or $S_Z^{-1'} = R_{x_Z}^{-1} R_z^{-1} R_x^{-1} T^{-1}$ ---- (3.13)

Again all the rotation transformation matrices viz. R_x, R_{x_Z}, R_y, R_{y_Z}, R_z are orthogonal matrices and their inverses are simply their transposes. This makes the computation of S_Z^{-1} very fast. At the end of this transformation process the surface patches approximating the surface of the pipe bend are ready for projection transformations.

Generation of the wireframe model is similar except that surface patches are not generated. Instead, selected points on the doubly curved surfaces are connected to generate polylines connecting the two hollow polygons representing the two end faces of the pipe bend. All the transformations including the inverse transformation are exactly similar to those for the surface model generation.

Surface model or wire frame model of vessel is the collection of surface patches or wire frame model of the cylinder. For a pipe it is the collection of surface patches for all the straight pipe pieces and pipe bends that make up the pipe. Once these models for all the pipes and vessels are generated, projection of these on various planes based on the view specified by the user is necessary to obtain the display. This involves projection of each of the patches in the surface model of the plant in case of surface model display and that of each line in case of wire frame model display.

3.2.4 Projection of Surface and Wireframe Models

Three views of the plant viz. plan, elevation and isometric are produced by the system using parallel orthogonal projections. The plane of projection, view reference point etc. needed for the projection transformations are decided by the system based on the view requested by the user.

Projection Transformations :- The plane of projection and direction of projection (which is normal to the plane of projection), can be specified by a vector originating from a reference point on the plane called view reference point, and the direction cosines of the normal Alternatively, along with the view reference point (VRP), (point A in figure 3 8), another point called center of view (point B in figure 3 8) can be specified. Geometric model (wireframe or surface model) of the plant is then transformed to a coordinate system with -Z-axis along AB with origin at A This can be achieved by using transformation matrix S similar to the one desribed earlier under generation of surface model After this transformation, the plane of projection is the Z-plane of the new coordinate system The projected view of the plant can now be obtained simply by neglecting the Z-coordinate in the transformed system and plotting the XY coordinates of each point with suitable scaling When a specific orientation of the view is desired e g in the plan view the +X-axis is required to be horizontal and towards right, a rotation transformation about the Z-axis of the transformed system is applied before the final projection on the Z-plane.

All the above transformations can be combined together in a single transformation matrix called the projection matrix, this matrix is solely dependent on the specification of vector AB and the orientation of the display which is typically specified by a vector called view up vector (VUP). This vector decides which direction should be vertical in the final display.

The projection transformation matrix is applied to the generated surface or wireframe model of the plant to produce the desired display. As shown in figure 3.9, the following view specifications are used to generate projection matrices for plan, elevation and isometric views

The planes of projection for plan and elevation views are y=a, and z=a

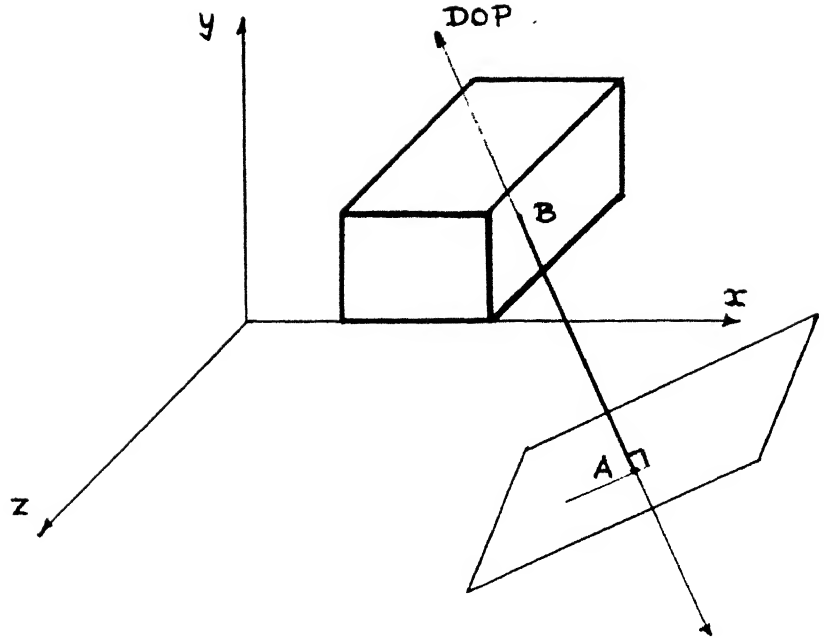


Figure 3.8 Projection Specifications.

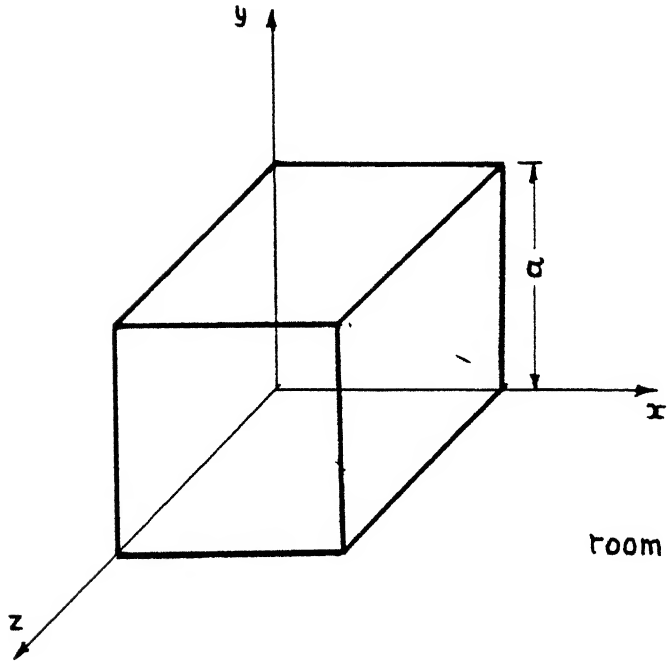


Figure 3.9 View Specifications.

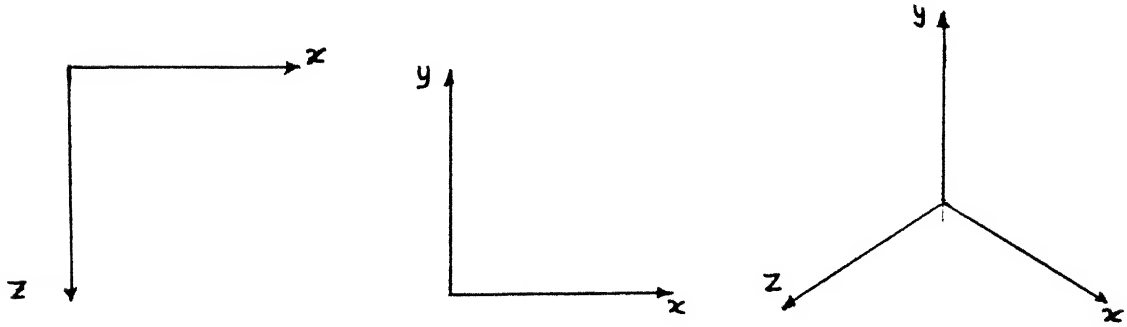


Figure 3.10 Orientation of the axes in the display.

respectively, and that for isometric view is a plane passing through the point (a,a,a), and equally inclined to the three principal planes.

The directions of projection (DOP) for plan, elevation and isometric views are (0,-1,0), (0,0,-1) and (1/√3,1/√3,1/√3) respectively; the corresponding view up vectors (VUP) are (0,0,-1), (0,1,0) and (0,1,0) This causes the orientation of the axes in the three views to be as shown in figure 3.10

The STARBASE graphics on HP-9000/360 series workstation has the facility to generate this matrix once the plane of the projection, view reference point and the orientation of the display are completely specifed

The coordinate system used by the graphics workstation for display, is a left-handed cartesian coordinate system, hence a transformation matrix to convert the right-handed cartesian coordinate system used in plant modelling, to the display coordinate system, is applied before projections are obtained This matrix is simply

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \quad \text{---- (3.14)}$$

If the view of the plant is required on a part of the screen, appropriate scaling and translation in 2-D is done after the projection transformations

3.2.5 Hidden Surface Removal

The above mentioned projection transformations apply to both the wireframe and surface model displays However, in surface model displays, it is necessary that surfaces obscured by other surfaces are removed from the display, for obtaining a more realistic display This is called hidden surface removal. It gives

an effect of relative depth of the object from the plane of projection. STARBASE graphics has a built in facility for hidden surface removal It uses the 'STRIP Z-buffer' algorithm for this It is a modification of the most commonly used Z-buffer algorithm which is explained in brief, below

Z- buffer algorithm - The basic idea is to maintain a depth value (called Z-value) for each pixel of the viewport on which the display is to be obtained Initially, all the pixels are assigned a large Z-value Then the projection of the surfaces is started The appropriate pixels are assigned attributes such as colour intensity etc depending upon the attributes of the surface For each pixel the depth of the point (corresponding to that pixel) on the surface, from the plane of projection (Z-value) is calculated and stored in the Z-buffer at the location corresponding to that pixel Then a second surface is taken up for projection The pixel attributes are changed depending on the surface attributes of this second surface, only if the Z-value calculated for the pixel from this surface is smaller than the current Z-value stored in the Z-buffer The Z-value in the Z-buffer is then updated to the new value. This happens if the point on the second surface represented by the pixel is not obscured by the corresponding point on the first surface If the point on the second surface is obscured by the point on the first surface, the pixel attributes are not changed and corresponding Z-buffer value is not updated. Thus, the obscured part of the surface fails to affect the pixel attributes and is not displayed on the screen. The next surface is then taken up and so on till all the surfaces are projected The end result is that only those parts of all the surfaces which are not obscured by other surfaces are displayed

This algorithm works very fast. The only disadvantage is that the Z-buffer requires a lot of storage space. To reduce the storage space requirement, the STRIP Z-buffer algorithm is used In this, the viewport is devided into a number of

vertical strips Each strip thus has lesser number of pixels and the Z-buffer required is smaller The picture corresponding to each strip is displayed sequentially After the display in one strip is completed, the same Z-buffer is reset and again used for display corresponding to the next strip

Shading and rendering is done by assigning attributes to each pixel depending on such factors as the distance of the point represented by the pixel from the light source, inclination of the surface normal at that point with the direction of the light source etc For the present work shading and rendering routines of the STARBASE graphics are used, only the position of the light source and the colour of the light are needed to be specified while calling these routines

CHAPTER 4

PIPE ROUTING

In this chapter, the three important tasks associated with the routing of pipes inside a plant unit are discussed, namely the detection of interference between a newly routed pipe and the existing vessels and pipes in the plant, graphic user assistance for manual routing of pipes and the implementation of automatic pipe routing algorithm

4.1 Interference Detection

Whenever any geometric changes are made by the user to the plant model, it is necessary to ensure, before accepting the changes, that it does not introduce any interference between the various components of the plant. As the vessels and pipes in the plant are stationary during operation of the plant, what is needed is a static interference check Two types of interference are possible in a plant model

(i) Hard Interference :- When the volume occupied by a component intersects the volume occupied by any other component, it is called Hard Interference The plant model in such a case is obviously geometrically inconsistent and not acceptable

(ii) Soft Interference:- The concept of soft interference comes from maintainability of the plant and some design considerations such as :

(a) Some equipment of the plant like heat exchangers, valves etc. need some additional volume apart from their assembled size, for maintainance and operation

such as removal of the tube bank of the heat exchanger, removal of an equipment such as a motor etc . In case of a valve, between the open and closed positions, the spindle and handle assembly sweeps a volume in addition to the valve body size

(b) Some volume in the plant must be free of any equipment for movement of maintainance equipment, walkways etc

(c) When two pipes are placed near each other, certain minimum clearance between the surfaces of the two is specified by the designer. This envelop around the pipe must be kept free of any equipment

In all the above cases, the additional volume that has to be kept free is called soft interference volume and can be allowed in a plant only conditionally For example if there is a pipe in the soft interference area of a heat exchanger then it is allowable only if removal of the pipe is feasible before removal of the tube bank of the heat exachanger Soft interference with the valve operating space as explained above, can never be allowed For checking soft interference, in cases (a) & (b) above, more detailed model of the equipment and the plant is needed Such soft interference is neglected in the present work For case (c) above, the clearance between pipes is specified and during interference detection the diameter of the pipe is increased to accomodate the 'soft' envelop of the pipe body

All the functions of the system allowing changes in the plant model such as addition/movement of vessels, manual and automatic routing of the pipes need interference detection. Each single component has to be checked with all the remaining components of the plant even for effecting the slightest change in the plant model Such rigorous demands of interference detection place very severe computational requirements, especially while autorouting For checking a component for interference, all the components of the plant, pipes and vessels are broken

down systematically into structures *part* representing the vessels and pipe pieces (straight pieces or pipe bends), as explained earlier Each single component of the plant thus represented by the structure part is then checked for interference with the component to be checked Thus the problem reduces to finding whether two given components of the plant represented by the structure *part* interfere or not. Two types of components of pipes and vessels are represented by the structure *part*

* Straight pipe pieces
* 90° pipe bends

Thus, interference between all the combinations of above two types has to be checked A most general method for detecting interference between above two types of parts is to generate the solid model from representation of the two parts and check if the intersection of these two models is an empty set If it is, then the parts do not interfere, else they do. However, for a pair of parts generally positioned/oriented in space, this procedure is a time consuming process

Interference detection, as it is done in most systems today, is an 'offline' task, i e the user specifies a pipe layout for a new pipe to be added to the plant and then the pipe model is fed to a rigorous interference detection routine such as the generalised one described above, which typically takes 8-10 hrs. and produces a report giving details of the locations where the pipe interferes with some other component of the plant The user then changes the route of the pipe in such a way as to suitably deal with these problem locations This method may be suitable for manual routing if at all such a detailed check is required. However, if the pipe routing is entirely left to the system as is the case with automatic pipe routing function of the present system, where the pipe is laid by the system from the start to end position specified by the user. an exponentially large number of

optional routes available have to be checked for interference 'on line' Hence, a very fast method of interference detection is required A detailed interference check as mentioned above is not suitable and it is not even necessary for the present work All that is needed is that if a part interferes with another it should never be approved It is quite acceptable to the present system that interference detection routines reject some parts that on a rigorous interference check as above would be revealed to be not interfering, provided the results of the checking procedure are produced fast enough In the end, the only effect this compromise would have on autorouting function is that in a very complex process plant it may miss some of the paths But a process plant of such a complexity is not practical

A practical and very fast method to detect interference is to determine the minimum distance between the axes of two parts, and to determine if at this point sufficient clearance is available between the surfaces of the two parts. If sufficient clearance is available then it is established that the parts do not interfere Else, it is established that they interfere The minimum axial clearance requirement can be adjusted to include soft interference envelope of each part This approach is very practical especially for regular geometric shapes, though the clearance can not be maintained exactly However, this does not matter much, since as compared to the plant unit size of 10 - 20 metres, the clearance of 100 mm or so is very small and variation of this clearance even by a factor of two is not at all harmful to the system performance in pipe routing functions. It is ensured that a minimum surface to surface clearance of 100 mm is maintained at all times To make the interference check even faster, the time to calculate minimum distance between two curves in space is saved by performing some early checks which can establish whether the parts interfere or not; e.g., if any of the

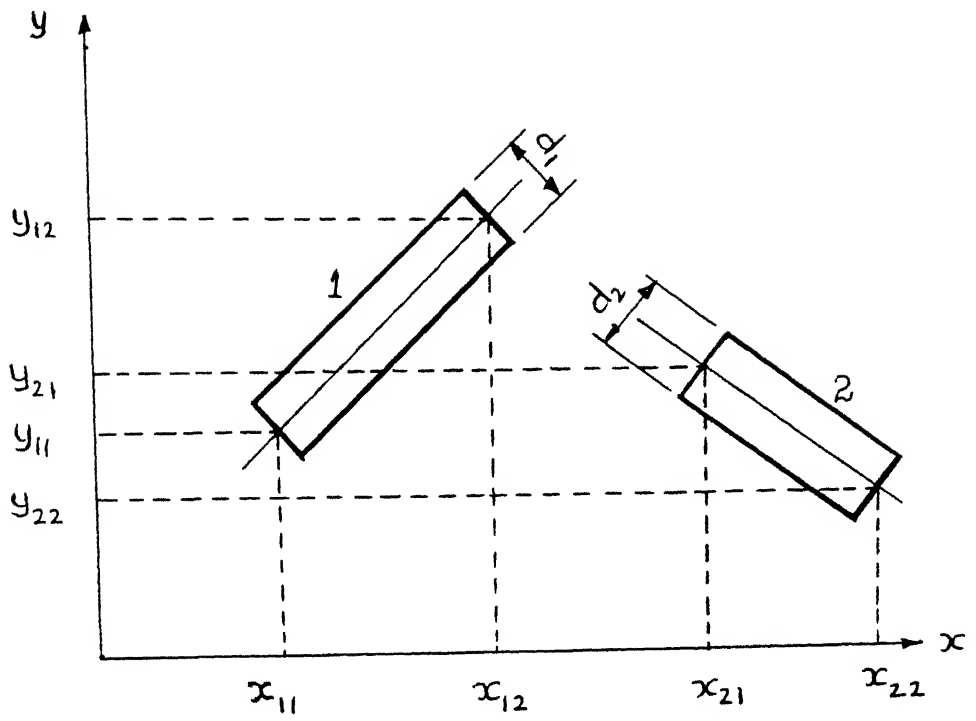


Figure 4.1 Interference Detection.

x, y or z extents of the parts are clear of each other by a distance greater than the minimum axial clearance required for no interference As shown in figure 4.1, if the criterion

$$x_{21} - x_{12} > \frac{d_1}{2} + \frac{d_2}{2} + \text{minimum clearance}$$

is satisfied, the x extents of components 1 & 2 are clear of each other Though the y extents are not clear, the parts definitely do not interfere. After such checks, the parts are categorised as straight pipe pieces, pipe bends etc and special cases such as straight parts with axes along principal directions are isolated and dealt with separately using special fast routines to give quicker results Only when such tests fail to produce any definite results regarding interference, the general procedures to determine the minimum axial distance available are invoked

The following algorithm is used to check interference between two components represented by the structure *part*. In the algorithm MINDIST is the minimum axial distance required to avoid interference, and SHORTDIST is the minimum axial distance available between the two components

Algorithm for interference detection between two components represented by structure *part* .

Accept a pair of components

Compare x,y and z extents of the two components.

If atleast one of the extents is clear,

return DONOTINTERFERE

If both components are straight pipe pieces ,

If both are parallel to principal directions,

Obtain SHORTDIST for straight pipe pieces in principal directions

Else

Obtain SHORTDIST for straight pipe pieces in general direction.

Else If both components are pipe bends,

If both bends are in parallel planes,

Obtain SHORTDIST for bends in parallel planes

Else

Obtain SHORTDIST for bends in perpendicular planes

Else

Obtain SHORTDIST for a straight pipe piece and a pipe bend

If SHORTDIST $\geq$ MINDIST

return DONOTINTERFERE

Else

return INTERFERE.

4.2 Manual Routing

To allow the user to specify a pipe layout manually the system is required to

* Pick up the pipe guiding points in 3-D space
* Check points entered for feasibility of pipe
* Build pipe model
* Check the built pipe for interference

(i) Picking points :- The system is required to pick points which are to serve as guide points for routing the pipe The problem is to pick points in 3-D space interactively through a 2-D display on screen. Following methods can be used to enter the points .

(a) Specifying a series of points by entering x, y and z coordinates of the points This is the easiest method for the system but obviously most difficult for the user because he has no graphic help from the system

(b) Displaying plan view of the plant and using a cursor to pick two coordinates of the point from the projected view and the third coordinate (the depth) being specified by the user seperately This method is slightly better than the first but still not satisfactory because the user has to keep track of the third coordinate of the points without any graphic help Also, this is very similar to working at drawing board with the plan view of the plant, since the designer uses a number of sectional plan views of the plant at various elevations. This method cannot be used by the system because a quick change of the sectional view as may be required by the user is not feasible

(c) Displaying two views (plan and elevation) of the plant, side by side in two viewports on the screen, and using two cursors separately in these viewports Combination of the positions of these two cursors can give the coordinates of a point in 3-D world coordinate space. This is the method used in the present work

Plan view of the plant is shown in left viewport and the elevation view is shown in the right viewport on the screen The user has the option to choose the mode of plant display, either as wireframe model or as surface model with hidden surfaces removed, which is slightly slower

Using such a display, 3-D points can be picked up with cursors of two types:

* Cursor in world coordinates :- A cursor in 3D space can be moved using three pairs of keys to control the movement in 3 directions. Projection of this marker in the two views of the plant as two cursors can be used to let the user know the current marker position. Simultaneously the marker position in world

coordinates can be displayed as x, y and z coordinates at the bottom of the display on the screen. However, this method is not as efficient as the one described below because at each position of the marker it has to be projected in the two views of the plant

* Cursor in device coordinates - Two cursors in device coordinates in separate viewports can be moved by the user using two pairs of keys and the combination of their positions on the screen can be related to the coordinates of a point in world coordinates using the scale of display. This method is used in the present work

The axes are as shown in figure 4 2 A cursor is shown in each viewport Each cursor can be moved in the respective viewports with normal cursor movement keys Vertical movement of the cursor in left viewport changes the z-coordinate and that of the cursor in the right viewport changes the y-coordinate; the direction of x axis is same in both views and the horizontal movement of any cursor moves the other cursor equally and changes the x-coordinate of the marker point in world coordinate space The user can switch the cursor control from one viewport to the other and can also change the speed of cursor movement.

The current position of the marker point is displayed continuously at the bottom of the screen as x, y and z coordinates Using the ENTER key a point is entered in an array of guide points for the pipe route. Checks to ensure that a pair of consecutive points entered always defines a line segment in one of the principal directions are incorporated in cursor movement control function and the user is not allowed to enter points arbitrarily

All the above gives the user absolute control over entering points to indicate the desired route In the end, the user enters the vessel connections of the intended pipe

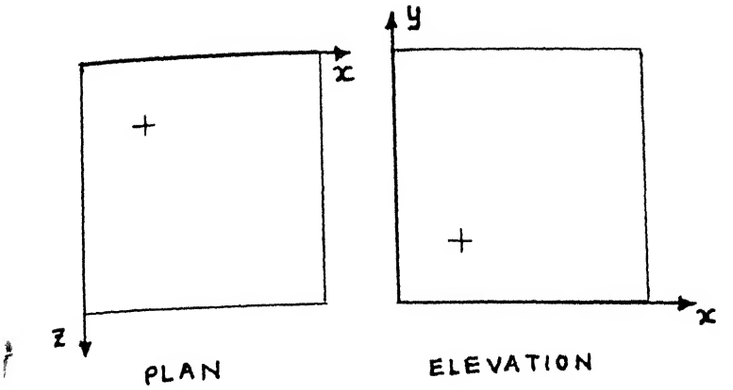


Figure 4.2 Picking points in 3-D.

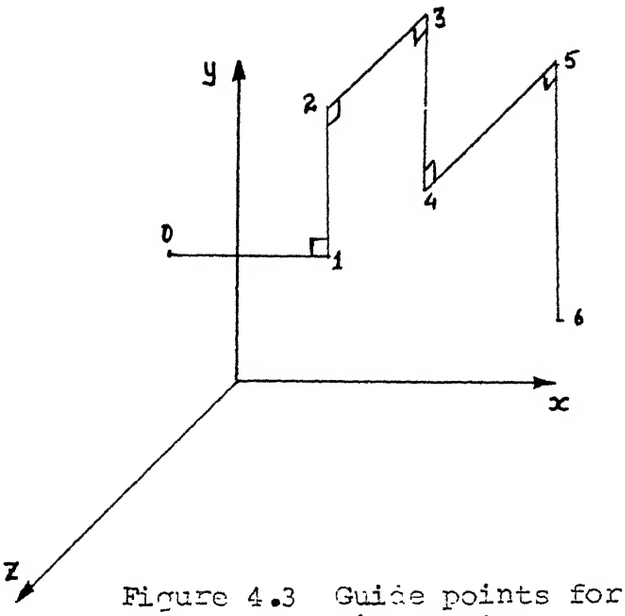


Figure 4.3 Guide points for the pipe route.

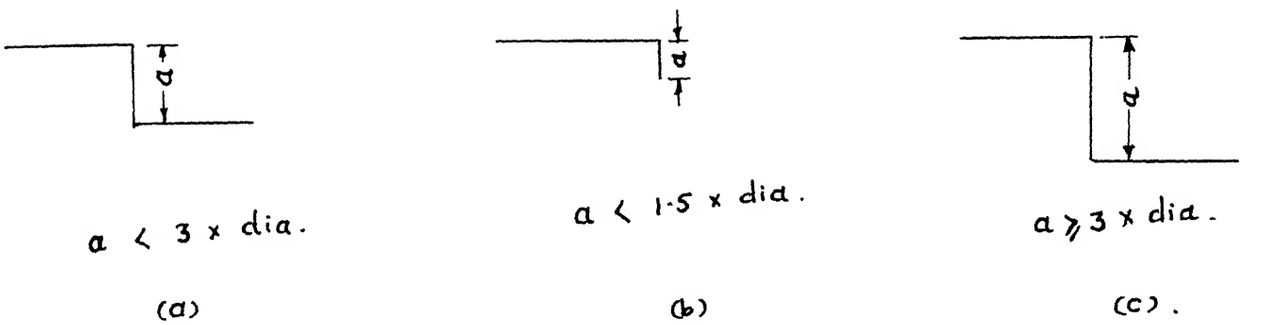


Figure 4.4 Feasibility of the pipe route.

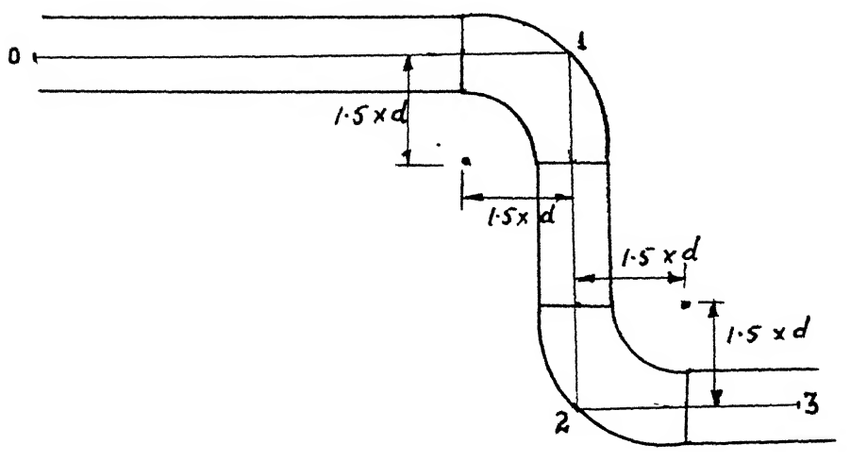


Figure 4.5 Generation of the pipe.

(ii) Checking feasibilty - When the array of points is entered by the user as shown in figure 4 3, it is necessary to check whether is is possible to build a pipe using these guide points The radius of the pipe bend used is 1 5 times the diameter of the pipe Hence as shown in figure 4.4, it is not possible to build a pipe for cases (a) & (b) and it is possible for case (c)

(iii) Building the pipe model - When the array of points is satisfactory from the viewpoint of feasibility, the pipe is built as shown in figure 4.5, by fitting straight pipe pieces and pipe bends. The structures representing these segments of the pipe are then built and linked in a linear linked list

(iv) Interference check and storage in plant model - Each piece of the pipe is then checked for interference with the whole plant If it interferes, it is discarded Else, the user has the option to add the pipe to plant model or discard it If the pipe is added to plant, vessel connections of the pipe and piperack occupation structres are updated suitably

4.3 Autorouting

Autorouting is the most important and most complex function of the system This function allows the user to specify two points in the plant room and ask the system to route a pipe of given diameter between the two points using minimum number of pipe bends The directions of the intended pipe at the two end points are also specified by the user At present a detailed geometric model of the structural members of the piperack is not included in the system, so the automatic pipe routing is not done using the piperack

The autorouting thus is essentially finding a path between two points in 3-d space avoiding obstacles A similar problem is that of moving a point in 3-d space avoiding obstacles such that the distance travelled by the point from the starting

point to the destination point is minimum.

However, the following major restrictions relating to pipe routing differentiate it from the problem of finding a minimum length of unobstructed path for a particle

(i) A particle can move along a line of insignificant width, whereas a pipe has a difinite diameter and occupies space

(ii) Path of a point can take sharp bends, whereas certain minimum space is required for changing the direction of the pipe as explained under the feasibility check in manual routing function This necessitates the availability of sufficient space between two consecutive bends

(iii) The pipe can use only 90° bends in this case, and the number of bends is required to be minimum possible

(iv) In case of pipe routing not only the source and the destination points but the directions of the pipe at these two points are also important

Thus the algorithm to be used for automatic routing of pipes has to incorporate all the above considerations such that if a path between the specified points is found, it must always be possible to lay out a pipe of given diameter along that path

The basic algorithm used in the present work for autorouting is the Dijkstra's algorithm for the shortest path The algorithm and its application to autorouting function are discussed in the following sections

4.3.1 Dijkstra's Algorithm for Shortest Path

This is basically an algorithm to find the minimum cost path (shortest path) between two nodes in a graph

Consider a graph as shown in figure 4.6 The n nodes (here n=8) of the graph are numbered 1 through n and an edge between any two nodes has some cost associated with it For non existing edges such as edge between nodes 3 and 8, an infinite cost is associated This indicates that there is no direct path between these two nodes Such a graph can be represented by a cost adjacency matrix COST[n][n] as shown in figure 4 7

In such a graph, it is desired to find the minimum cost path from a source node to the destination node The algorithm described below achieves this by entering node numbers in a set S in the nondecreasing order of distances of the nodes from the source node, the distance of a node being defined as the cost incurred while travelling from the source node to that node. These distances of all the nodes from the source node are maintained in an array DIST [n] Thus, all the nodes whose distances from the source node are less than the distance of the destination node from the source node are added to the set S and when finally the destination node is added to the set the procedure stops Each time a node is added to the set S distances of the nodes directly connected to this node may be reduced These updates are made before searching the graph for the next nearest node It can be proved that the shortest path originating from the source node to a freshly added node to the set S, always passes through the nodes in set S only Thus the least cost paths to various nodes i which have associated DIST[i] values lesser than that associated with the path from the source to the destination node are generated before the least cost path to the destination node is generated

The algorithm is listed below and its working on the graph of figure 4 6 is shown in the table 4.1

The algorithm assumes that .

(i) n nodes of the graph are serially numbered 1 through n

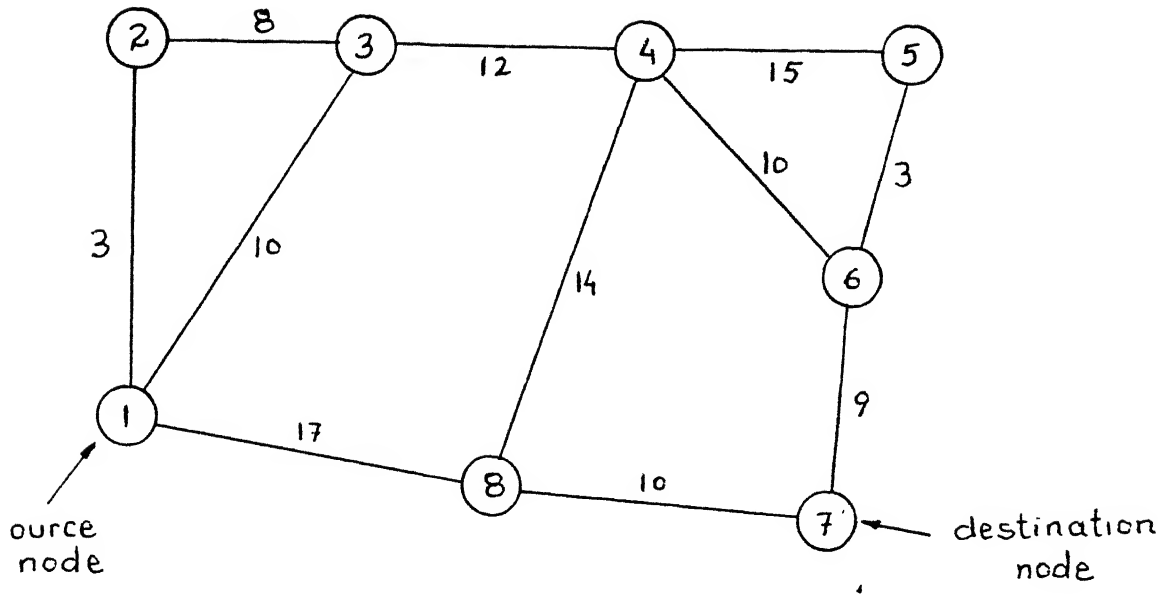


Figure 4.6 A graph with 8 nodes.

NODES	1	2	3	4	5	6	7	8
1	0	3	10	–	–	–	–	17
2		0	8	–	–	–	–	–
3			0	12	–	–	–	–
4				0	15	10	–	14
5					0	3	–	–
6						0	9	–
7							0	10
8								0

Figure 4.7 Representation of a graph as a matrix.

(ii) the graph itself is represented as a cost adjacency matrix (figure 4.7) with COST [i][j] being the cost associated with edge(i,j) COST[i][j] is set to a large value (infinity) in case edge(i,j) does not exist COST[i][j] for i=j may be stored as any non negative value

(iii) the distances from the source node, i.e. the costs associated with the path between the source node to any node i are stored in DIST[i]

(iv) set S is maintained as an array S[n] with S[i] =0 if node i is not in set S and S[i]=1 if it is

Dijkstra's Algorithm for Shortest Path :-

```
/* source node is v     DIST[v] is set to 0     number of nodes is n */

unsigned char S[n],    short COST[n][n].     integer u, v, n, num, i, w ;

for i = 1 to n  do   /* initialise set S to empty */

        S[i] = 0 , DIST[i] = COST[v][i] ,

repeat

S[v] = 1 , DIST[v] = 0 ,    /* put node v in set S */

for  num = 2 to n-1 do

        choose u such that DIST[u] = min { DIST[w] ) for S[w] = 0

        S[u] = 1     /* put node u in set S */

        for all w with S[w] = 0  do    /* update DIST array  */

                DIST[w] = minimum ( DIST[w] , DIST [u] + COST[u][w] )

        repeat

 repeat

stop
```

Sequence of node addition to set S	Cost of Shortest Path from node 1 after addition of each node to set S							
	1	2	3	4	5	6	7	8
1	0	3	10	--	--	--	--	17
2	0	3	10	--	--	--	--	17
3	0	3	10	22	--	--	--	17
8	0	3	10	22	--	31	27	17
4	0	3	10	22	37	31	27	17
7	0	3	10	22	37	31	27	17

Source node no.:- 1 destination node no.- 7

Table 4.1 Illustration of Dijkstra's Algorithm

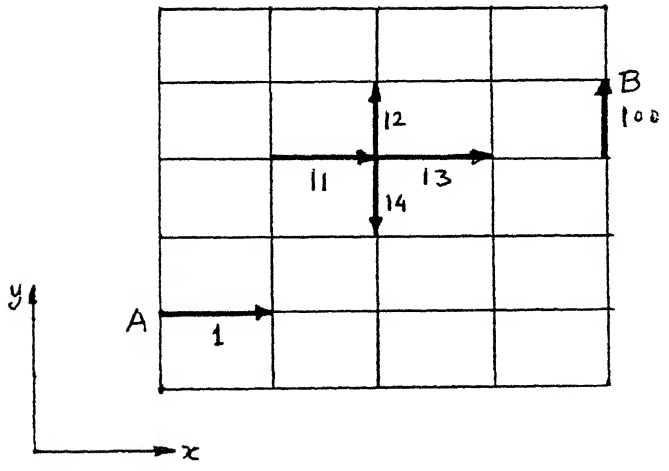


Figure 4.3 Mesh for autorouting in 2-D.

As can be seen from the algorithm, the procedure first puts the source node in S, updates the DIST array for all nodes not in set S by assigning minimum of DIST[i] and DIST[1] + COST[1][i] to DIST[i] It then finds the minimum from the array DIST, out of all the entries i for which S[i] = 0, and enters the corresponding node number, say j to set S It updates the DIST array for all nodes i not in set S, using minimum of DIST[i] and DIST[j]+COST[j][i] and enters the next minimum to set S and so on, till the destination node is added to set S For example in the graph of figure 4 6 if source node is 1 and destination node is 7, the procedure works as shown table 4 1 In the table the unfilled entries denote infinity The distance associated with the destiantion node at the time of its entry to set S is the cost of the shortest path from source to destination node The nodes in the example are added to set S in the sequence 1-2-3-8-4-7 and the shortest path cost from node 1 node 7 is determined as 27.

However, the shortest path itself is not produced by the algorithm For getting the actual path in terms of a sequence of nodes on the path, a slight imporvement over the above algorithm is needed Each time the DIST[i] i.e., the distance associated with node i is updated due to the addition of node j to the set S, the node number of this freshly added node j is stored in array PATH[n] at location PATH[i] Thus for each node i its parent node (node j) which has caused reduction in DIST[i], is available When the goal node is reached it is then easy to backtrack the path from destination node to the source node using the array PATH.

4.3.2 Application of Dijkstra's Algorithm for Automatic Routing of Pipes

The problem of finding a path in 3-d space between two points with some geometric constraints is similar to finding the shortest path in a graph with the cost constraint The conversion of the physical problem of autorouting. to a graph

search is explained below in 2-d space for simplicity The same procedure can be extended to 3-d space

Consider two points A and B between which a pipe of diameter d is to be routed avoiding obstacles and using minimum possible number of 90° bends only (figure 4 8) In addition to the coordinates of the two points, the direction of the pipe at the two end points is also important The two directions are specified as say +x at point A and +y at point B A mesh of rectangles is generated as shown with the sides of rectangles greater than or equal to three times the diameter of the pipe to be routed The size of the rectangles is adjusted so that the two points A and B fall on the mesh points. As the turning radius of the pipe i e, radius of the pipe bend axis is 1.5 times the diameter of the pipe, a minimum rectangle side of three times the diameter is necessary for the pipe route to be feasible

Now, with each line segment of the mesh are associated two nodes: One representing the positive direction and the other representing the negative direction All such interconnected nodes form a graph Depending on the coordinates of the points A and B and the directions associated with them, the source and destination nodes are identified Thus in figure 4.8, segment 1 is the source and segment 100 is the destination node The extension of the mesh around the points is guided by the pipe directions at points A and B and the general restriction on the space available for pipe layout. This in fact, will indirectly bring about minimisation of the length of the pipe, because lesser volume for searching the path will be available

Now, the problem of autorouting from point A to B reduces to finding a sequence of segments in figure 4 8. along which a pipe of a given diameter can be routed avoiding all obstacles Thus, in this sequence only those nodes are allowed

which do not interfere with any equipment or pipe already existing in the plant There is a further restriction that the path should have minimum possible bends Thus, each segment of the mesh is used as two nodes of a graph one in each direction For applying Dijkstra's algorithm for finding the minimum cost path (path with minimum number of bends in this case) the following is needed

(i) Representation of the mesh as a graph by a cost adjacency matrix which contains the cost of travel from each node to every other node in the graph

(ii) A list of nodes that cannot be included in the path due to their interference with other pipes and equipment in the plant, so as to make sure that they are not included in the path

(iii) Establishment of the optimzation criterion namely, the 'cost' of a path

To implement the algorithm the following methodolgy is adopted which incorporates all the above requirements

(i) The area between the two points is specified and some additional area around the points is divided into a mesh.

(ii) Each node (two nos. corresponding to each segment of the mesh) is uniquely numbered.

(iii) From each node three child nodes can be generated, i.e., from each node we can reach only three other nodes with a finite cost For example as shown figure 4 8, node number 11 which is in +x direction has three child nodes 12,13 ,14 As the algorithm is required to seek the path with minimum number of bends, a cost of 1 unit is associated with a bend and a cost of 0 units is associated with a straight pipe piece Thus COST[11][13] =0 and COST[11][12]=COST[11][14]=1. There are no edges between node 11 and all nodes other than its child nodes So COST[11][j] for all j not equal to 12, 13 or14, is infinity If it is not possible to

move from node 11 to any of the child nodes say 12 due to an obstacle, the cost associated with edge(11,12) ,i e ,COST[11][12] is made infinity. This is checked using interference detection routines For this, pipe pieces are generated using coordinates of the endpoints of the node Typically, a pipe piece of oversize diameter with the axis along the node under consideration is checked for interference with the rest of the equipment and pipes in the plant If the interference check is negative the DIST entry for the node is made infinity so that this node is never added to the set S

(iii) An array DIST[n] is maintained in which cost associated with the path from the source node to a node i is stored at DIST[i]

(iv) An array PATH[n] containing node numbers of the parent node of each node is maintained This is necessary to determine the shortest path when the destination node is reached, as explained earlier

The modules required for the above, and the implementation of the algorithm are discussed later in section 4 3 4. Armed with the above, the procedure based on Dijkstra's algorithm for the shortest path proceeds as follows .

(i) It identifies the source and the destination nodes from the coordinates of the two endpoints and the corresponding directions associated with them.

(ii) It initialises set S to empty and DIST[i] for i = 1 through n, to infinity

(iii) It adds the source node to set S, (e g node no 1 in figure 4.8) and generates the child nodes of node 1

(iv) It updates the DIST entries of the freshly generated nodes only, because the DIST entries of all the nodes which are not freshly generated are not affected by the addition of the previous node to set S Thus if j is a child node of i, the update for node j is

DIST[j] = minimum (DIST[j] , DIST[i] + COST[i][j])

if nodes i and j are collinear then COST[i][j] = 0

if nodes i and j are at right angles then COST[i][j] = 1

if node j does not clear interference check then COST[i][j] = infinity

(v) For all the child nodes j whose DIST entries are updated due to a parent node i, the node number i is stored in the array PATH at PATH[j]. Thus at any time the array PATH contains the node number of a node just preceding it in the shortest path from the source node

(vi) It determines the minimum entry of the DIST array out of all the nodes that are not in the set S, and adds this node number corresponding to the minimum entry, to the set S

(vii) It repeats steps (iv) to (vi) till the destination node is added to set S

(viii) It determines the parent node of the destination node, its grandparent node and so on till the path is traced back to the source node This is done by using the array PATH

At the end of the procedure, a sequence of nodes (directed line segments in the mesh of figure 4.8) is available The two endpoints of each segment in this sequence can now be obtained and an array of points can be generated which serve as guide points for routing the pipe This path generated by the procedure is obviously a path with minimum number of bends, because the procedure works out a minimum cost path, and with each bend a finite cost of 1 unit is associated whereas a cost of 0 units is associated with a straight pipe piece After the array of guide points is available, generation of the pipe model is exactly the same as described under manual routing

For routing in 3-D space, exactly the same procedure is followed except that the search space (volume) is divided into a mesh of rectangular blocks rather than rectangles, with the edges of the blocks representing the nodes, two nodes per edge for the positive and negative directions The number of child nodes of a node in 3-D space is 5 instead of 3

4.3.3 Evaluation of Dijkstra's Algorithm Applied to Autorouting

The main problem with the implementation of this algorithm for autorouting of pipes arises due to the large number of nodes involved If the distance between the two endpoints of the intended pipe is very large as compared to the pipe diameter, the number of nodes in the graph becomes too large. For instance, for a pipe of 50 mm diameter the mesh size of 150 mm is suitable If the x, y, and z coordinates of the two endpoints differ by say 6000 mm each, then the number of node in the graph will be

nodes in each of the x, y, and z directions $= 2 * (6000 / 150)^3 = 128000$

total nodes (n) = 3 * 128000 = 384000

Following is a discussion of the problems due to such a large number of nodes and the methods adopted in the present work to solve them

(i) Storage of the graph as a cost adjacency matrix is a major problem when the number of nodes is large A graph of n nodes can be represented by a cost adjacency matrix COST[n][n] For a large value of n it is impossible to store this huge matrix To overcome this, use is made of the fact that COST[i][j] is infinity if node j is not a child node or parent node of node i Thus, COST[i][j] value is required in computation when node i is added to set S, the child nodes j of node i are generated and for each such j COST[i][j] is set to 0 or 1 depending on whether

node j is collinear with node i or perpendicular to it

(ii) All the nodes of the graph which when included in the path for the pipe, cause the pipe to interfere with other equipment and pipes of the plant, have to be restrained from entering the set S For this the costs associated with these nodes have to be made infinity Interference checking for such a large number of nodes needs a lot of time So instead of checking all the nodes for interference and storing the cost of interfering nodes as infinity, use is made of the fact that a node can be added to the set S only after it is generated as a child node of some node already in set S Thus as soon as a node is generated it is checked for interference and a tag corresponding to this node is set to ON indicating that the node has been checked for interference If the node interferes, its DIST value is set to infinity When this node is generated again as a child node of some other node it will not be checked for interference because the interference tag is ON This ensures that only those nodes which are likely to be included in the set S, are checked for interference, and that these nodes are checked only once This saves a considerable amount of time

(iii) Searching for the minimum entry in the array DIST is a costly operation for a large array, especially when it is carried out each time a node is added to set S To avoid searching the whole array to find the minimum, use is made of the fact that the nodes are added to the set S in a non-decreasing order of their distances from the source node, i.e. their DIST values; the added node is then removed from the DIST array. So, while searching for a minimum in the DIST array it is guaranteed that all the entries in the array are either equal to or greater than the previous minimum value Thus, instead of searching the whole array, the search is stopped as soon as a value equal to the previous minimum is found If no such entry is found, the search is continued to the end of the array and the

required minimum is found by simple search. This considerably reduces the search time because in most cases the complete array is not needed to be scanned

4.3.4 Modules Developed for Implementation

This subsection shows the breakup of the overall work of conversion of the pipe routing problem to a minimum cost path finding problem, and its solution using Dijkstra's shortest path algorithm. A listing of the role of each module in the overall problem is given below Each of the modules listed below is made up of one or more functions

(i) Given two endpoints and the corresponding directions of the intended pipe, establishing the search volume for the path, generation of the mesh of rectangular blocks and adjusting the number of nodes depending on the user specified level of intensity of search The mesh is adjusted such that the two endpoints lie on the mesh points.

(ii) Establishing a unique identification number for each node of the graph to facilitate reference to the nodes simply by a node number. Each edge of the elemental rectangular block of the mesh represents two nodes of the graph, one in either direction All the nodes in positive direction are numbered as triplets of x, y and z direction nodes originating from each mesh point in a definite sequence and then the same pattern is followed for negative direction nodes (figure 4 9) Here m is the total number of positive direction nodes in the graph

(iii) Identification of the source and destination nodes, generation of the coordinates of the endpoints of any node identified by the node number.

(iv) Given a node number, generation of its child nodes, checking interference, updating the DIST values and searching the minimum entry in the DIST

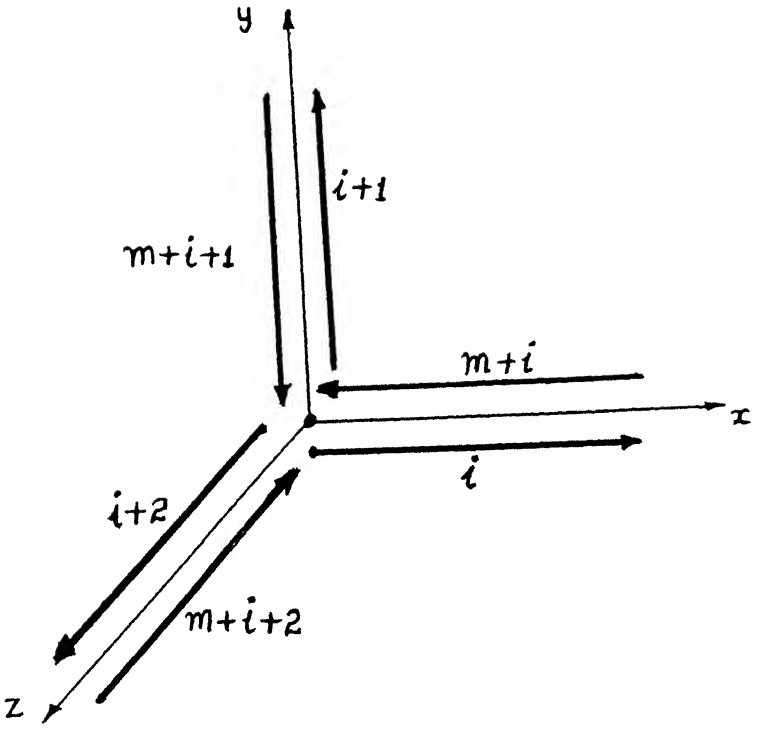


Figure 4.9 Numbering of nodes for autorouting.

array

(v) Generation of the minimum bends path when the goal node is reached, using the parent node entries in the array PATH, establishing the guide points for the pipe, and building the pipe model

4.4 Conclusion

A complete pipe routing function is thus incorporated in the system The manual and automatic routing functions make extensive use of the interference detection module and other supporting functions After the placement of vessels in the plant is complete, the pipe routing functions can be used to layout the pipes Initially the pipes can be routed with the automatic routing function, using different intensities of search, and if, even by using the most intense search, the system fails to find a path for the pipe, or if the path generated is not suitable for any reasons, the manual routing function can be used to layout the pipes manually

CHAPTER 5

RESULTS, DISCUSSION AND SCOPE FOR FURTHER WORK

The software developed in the present work, though is a long way off from a commercially usable system envisaged by Engineers India Ltd mentioned earlier in chapter 1, it is neverthless a first step towards that goal

One of the requirements of the overall system was automatic routing of the pipes This is an important part of the present work and to lay more stress on this some of the lesser important functions like geometric modelling of vessels which can be represented as combination of cylinders and spheres using constructive solid geometry, were not considered Within the limitations and assumptions explained in chapter 1, all basic functions required of such a system are built The system is very modular and is adaptable to improvement/expansion on all fronts

For example, if the facility to model a spherical vessel is to be added to the system it can be easily done by including routines and modifications relating to the geometric modelling of a sphere. This will include representation of the spherical vessel, modules for data conversion to generate surface model of a sphere, interference detection of a spherical vessel with all the other equipment and pipes in the plant etc These modules can simply be added to the exising system with only slight changes to the controlling routines. As long as interference detection routines are suitably improved to accomodate the new types of equipment added to the plant model, the manual and automatic pipe routing

functions remain unaffected

If facilities for including vessels that can be modelled as a combination of cylinders and spheres etc are to be added to the system, the interference detection routines will remain unaffected because each basic component of such a vessel can be separately checked for interference The manual and automatic pipe routing functions will again remain unaffected.

In case, non-geometric attributes are to be added to the pipe model such as material code, pressure rating etc , such information can be stored in the pipe representation either in the structure part representing a pipe piece or in a separate structure which can be linked to the pipe pointer. Such an improved version of the pipe model can then be used to communicate with certain pipe design/analysis packages like those for thermal stress analysis of pipes This change will naturally require no alterations to the plant manipulation, display and routing functions Similar changes in the representation can be made and these can be used for automating documentation such as bill of materials etc. These and a lot of other changes are possible without changing the basic structure of the present system

As regards the display function of the system all the basic things needed, like conversion of the data to generate surface model, transformations etc. are included in the system A lot of facilities which would make the system more user friendly, can be added on without any problem. Such facilities may include part views of the plant, arbitrarily selected sectional views etc. For production of all these displays only the view specifications like view reference point, plane of projection etc. have to be specified The same display routines as available in the present system, can then be used to obtain the desired display

Manual routing function of the system performs well and can be improved

upon by using facilities such as mouse for picking points etc

The time required to find a path with minimum number of bends, is an important factor for the success of the automatic pipe routing function As explained earlier, the search time increases rapidly with the increase in the number of nodes (n) in the graph. Hence at every stage in the program, effort is made to reduce the computational time As explained under interference detection which is a major time consuming process, a considerable time is saved by avoiding unnecessary exact calculations Other operations which are repeatedly required, like finding minimum from a large array etc , are made efficient as explained earlier As a result of the above simplifications the automatic routing function is very fast Table 5 1 gives an idea of the actual time taken for automatic routing of a pipe, for some representative cases The time shown in the table for each case will vary with other factors like complexity of the plant, and the results only give a general idea

The time taken by the algorithm used in the present work, for automatic routing of the pipes between given two points depends primarily on the distance between the two points, diameter of the pipe and the complexity of the plant. If the distance between the two points is large, the number of alternative routes possible is large, and the time required to search a suitable minimum bends route is more In terms of the algorithm used, this means that the number of nodes in the graph to be searched is large In case of pipes of smaller diameter, again the number of nodes of the graph is large and the search time is more Finally, the time required depends on the complexity of the plant, more the number of obstacles and larger the obstacles, more is the time taken to find an unobstructed route, because the interference checks needed are more and costlier in time However this effect is less pronounced than that due to the increase in the number of

S No	dia (mm)	dx (mm)	dy (mm)	dz (mm)	no of nodes	no of bends	time required
1	100	861	782	704	2058	3	6 sec
2	100	2269	1799	2034	7920	3	96 sec
3	100	2817	0	0	2100	0	1 sec
4	100	3052	0	0	2250	2	2 sec
5	100	2426	2269	0	4680	3	40 sec
6	200	4538	6729	7902	18432	2	80 sec
7	200	4930	1565	1800	3528	4	15 sec
8	200	2739	2895	3365	4860	3	12 sec
9	200	1956	6573	4930	8736	3	57 sec
10	200	0	5477	4930	5460	2	6 sec

Table 5-1. CASE STUDIES (AUTOROUTING)

nodes

As such, a substantial reduction in the autorouting time can be effected by choosing a coarser mesh which reduces the number of nodes in the graph. However, this may lead to missing some of the routes which might have been found with a finer mesh But if the obstacles are sparse this will not create any problem Thus the long distance pipe routing using a coarser mesh can be done first when the plant is sparsely populated Addition of the pipes will increase the complexity of the plant Then the shorter distance pipes can be laid out using a finer mesh so that the complexity of the plant does not severely affect the chances of finding a suitable path Also searching a path for a larger diameter pipe is difficult in a complex plant, so the larger diameter pipes should be laid out in the earlier stages of pipe routing when the plant is comparatively sparsely populated

All this is consistent with the actual pipe laying out process followed by the designers who first layout larger diameter and long distance piping, and then go for smaller diameter short distance piping

The system in its prsesnt form, offers the option to the user to decide the intensity of the search Three options are available to the user, low, medium and high intensity searches Based on the option selected by the user an appropiate mesh is generated and used for searching the path; the more intense the search demanded, the finer is the mesh used A high intesity search naturally takes comparatively more time and should be used if the plant is densely populated and the user expects difficulty in finding the path In the initial stages of the pipe routing when the plant is sparsely populated, low and medium intensity search options should be used.

The autorouting function of the system suffers from the drawback that in

case of an extremely complex plant, it may not necessarily find a feasible path However, considering the speed with which a path is found, the system is better than existing commercial software which in fact do not have any true automatic pipe routing facilities The maximum that is available is an interference detection function that operates in batch mode So, even if it relieves the user of the responsibility to check the geometric consistency of the plant model, finding the pipe route and the necessary changes to avoid the detected interference locations, are left to the user This is closer to the manual routing function of the present system, whereas the autorouting function of the present software completely automates the pipe routing All that the user is needed to do is to specify the two endpoints and the corresponding directions of the intended pipe. The system then finds a path with minimum number of bends and builds the pipe model ready to be added to the plant model, thereby relieving the user from the responsibilty to check geometric feasibility of the pipe

In the present system, a most general case of two points in 3-D space is considered The system does not detect simpler cases of pairs of points which can be connected by a simple L-shaped or U-shaped pipe layouts in a single plane; it goes through the regular path finding procedure implemented for the most general case An early detection of such cases will improve the autorouting function of the system If any of the differences between the x, y and z coordinates of the two points is less than three times the diameter of the intended pipe, but not equal to 0, the present algorithm can not be used for autorouting because a suitable mesh can not be generated The system fails to autoroute such a pipe even though a simple route may actually be possible. This drawback can be overcome by building ability in the system to identify a suitable intermediate point so that routing of the pipe from this point to the two user specified points does not pose the above

difficulty, and then the pipe may be routed via such a point

A major improvement in the autorouting function of the system can be brought about by modifying it to allow the shifting of mesh Some paths which may be missed using a particular mesh can be found by this method

Apart from the above improvements to the present form of the system, a lot can be done to make it a commercial system One of the immediate things that can be done is the establishment of a database of the standard pipe and equipment used in the process plant, and building abilities in the system to extract data from such a database The data of standard valves, heat exchangers, centrifuges etc. can be stored in a database, and can be refered by the system to extract relevant data for geometric modelling of such equipment. Generation of piping isometrics and bill of materials etc can be done easily.

Within the restrictions explained in Chapter 1 the system performs very well A set of photographs demonstrating the various functions of the system is attached.

Pictures 5.1 through 5.3 demonstrate the movement of a vessel An already existing vessel is being moved to a new location and orientation As shown in the picture 5 3 the original vessel is deleted from the plant model and the new position is stored in its place.

Pictures 5.4 and 5.5 show the isometric views of the plant as a wireframe model and as a surface model with the hidden surfaces removed, respectively

Pictures 5 6 through 5.8 demonstrate the manual routing function of the system In picture 5.6 a pipe is shown as an outline connecting the guide points entered by the user. Picture 5.7 shows the pipe built by the system after the feasibility and interference checks are carried out This pipe model is then added

to the plant model Picture 5 8 shows the same pipe in isometric view

Pictures 5 9 through 5 16 demonstrate the automatic pipe routing function In these pictures, the two endpoints and the corresponding directions of the intended pipe are kept the same, and the system is repeatedly asked to route the pipe, each time gradually increasing the obstacles in the path of the pipe generated. As can be seen in picture 5 9 the pipe route is simple because there are no obstacles in the path In figures 5 10 through 5 16 the system changes the route of the pipe to avoid the obstacles introduced in the simple path, by increasing the number of bends, if necessary

REFERENCES

(1) Mandalagiri S Chandrasekhar and Melvin A Breuer, *"Optimal Routing of Two Rectangular Blocks"*, -- IEEE transactions on Computer Aided Design, volume 8, number 4, April 1989 -- pp 413 - 430

(2) Youn-Long Lin, Yu-Chin Hsu and Fur-Shing Tsai, *"Hybrid Routing"*, -- IEEE transactions on CAD of Integrated Circuits and Systems, volume 9, number 2, February 1990 -- pp 151 - 157

(3) J P Cohoon and P L. Heck, *"BEAVER A Computational-geometry-based Tool for Switchbox Routing"*, -- IEEE transactions on Computer Aided Design, volume 7, June 1988 -- pp 684 - 697

(4) Ellis Horowitz and Sartaj Sahni, *"Fundamentals of Computer Algorithms"*, Galgotia Publications, New Delhi, September 1984

(5) Michael E Mortenson, *"Geometric Modelling"*, John Wiley & Sons, New York, March 1985

(6) James D Foley and Andries Van Dam, *"Fundamentals of Interactive Computer Graphics"*, Addison-Wesley Publishing Company, Reading, Massachusetts

(7) STARBASE graphics manual

APPENDIX

Calculation of Minimum Axial Distance Available Between Straight Pipe Pieces and Pipe Bends for Interference Detection Routines.

As a basis of interference detection calculations the minimum axial distance available between the two components represented by the c structure part, has to be determined for all combinations of straight pipe pieces and pipe bends If CLEAR is the minimum surface to surface clearance desired between the components, and if SHORTDIST is the shortest axial distance available, then for all cases, if SHORTDIST is greater than the minimum axial distance required to avoid interference (MINDIST), then the components do not interfere, else they do.

$$MINDIST = \frac{d_1}{2} + \frac{d_2}{2} + CLEAR \qquad \text{--- (A.1)}$$

where d_1 and d_2 are the two pipe diameters.

Still, the task of finding the minimum axial distance available between any two given components represented by the structure part (SHORTDIST) remains . Each section below deals with a different combination viz two straight pipe pieces, two pipe bends and a straight pipe piece and a pipe bend

Let AB and CD be the axes (line segments for straight pipe pieces and arc of a circle for pipe bends)

A.1 Minimum Axial Distance Available Between Straight Pipe Pieces

The following two cases need consideration

(i) Both AB and CD are in Principal Directions

For this, again two possibilities arise

(a) AB and CD are parallel to each other - In such a case let AB and CD be

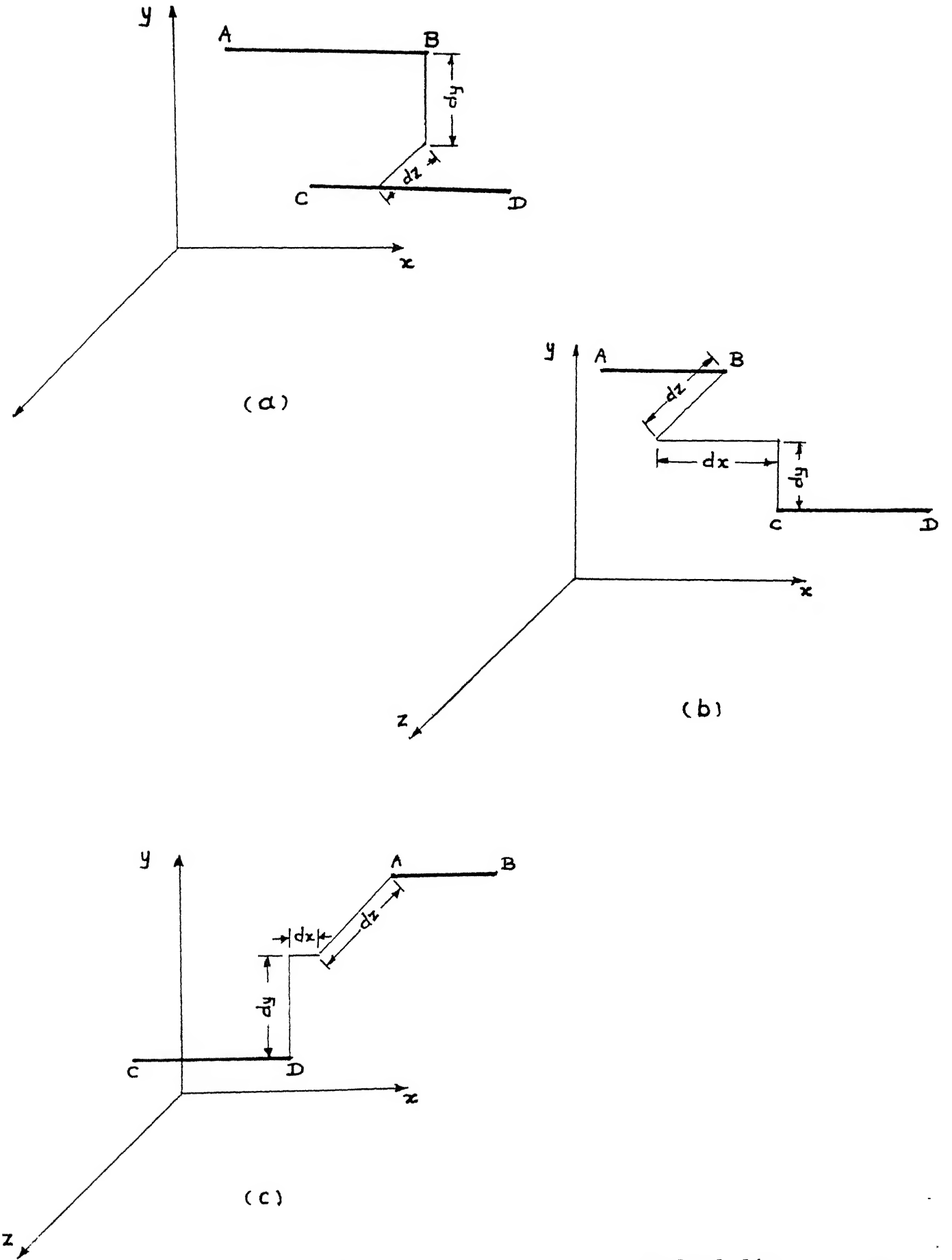


Figure A-1 Shortest distance between parallel line segments in principal directions.

parallel to x axis If x extents of AB and CD overlap (figure A-1 (a))

i e $\max(x_A , x_B) > \min(x_C , x_D)$ or $\min(x_A , x_B) < \max(x_C , x_D)$

then SHORTDIST = $(dy^2 + dz^2)^{1/2} = ((y_B - y_C)^2 + (z_B - z_C)^2)^{1/2}$ --- (A.2)

If the x extents do not overlap

then SHORTDIST = $(dx^2 + dy^2 + dz^2)$

= length (BC) for $\max(x_A , x_B) < \min(x_C , x_D)$ (figure A-1 (b))

= length (AD) for $\min(x_A , x_B) > \max(x_C , x_D)$ (figure A-1 (c)) --- (A 3)

For AB and CD parallel to y or z axis the treatment is similar

(b) AB and CD are perpendicular to each other :- Let AB be parallel to x axis and CD be parallel to y axis. It has already been established that none of the x , y or z extents are separated by the minimum axial clearance desired. Hence for the case shown in figure A-2 (a) where both x and y extents of AB and CD overlap the components definitely interfere No further investigation is necessary

For the case in figure A-2 (b)

SHORTDIST = $(dx^2 + dz^2)^{1/2} = ((x_B - x_C)^2 + (z_B - z_C)^2)^{1/2}$ --- (A 4)

For figure A-2 (c)

SHORTDIST = $(dy^2 + dz^2)^{1/2} = ((y_B - y_C)^2 + (z_B - z_C)^2)^{1/2}$ --- (A 5)

For figure A-2 (d)

SHORTDIST = min(length (AC) , length (AD), length (BC), length(BD)) --- (A 6)

For cases where AB and CD are in x and z or y and z directions similar treatment is applicable

(ii) Atleast one of the Segments AB and CD is not Parallel to any Principal Direction

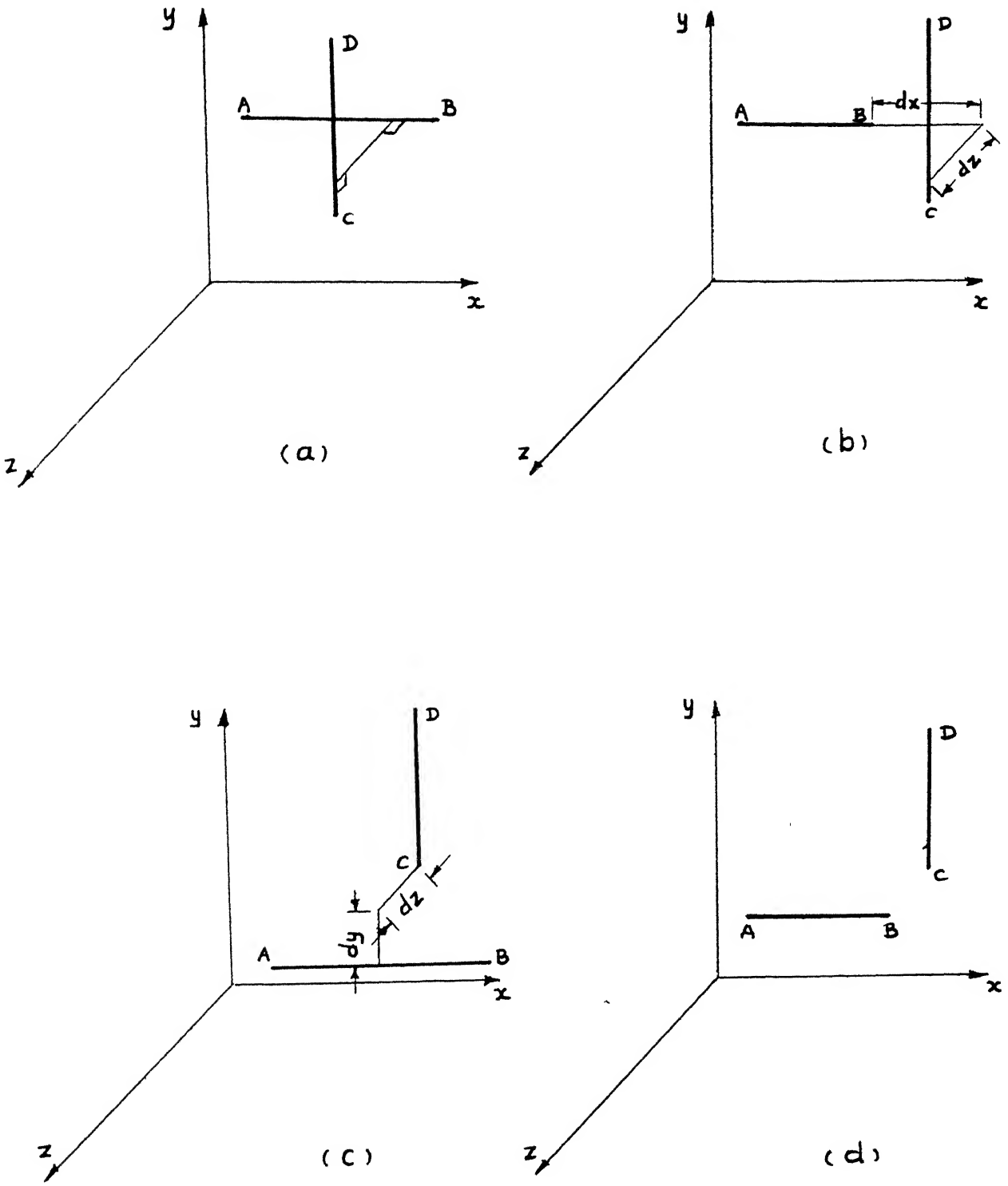


Figure A-2 Shortest distance between perpendicular line segments in principal directions.

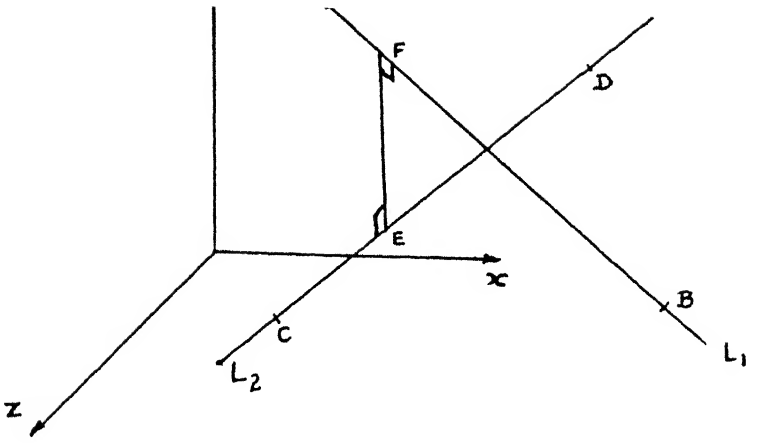
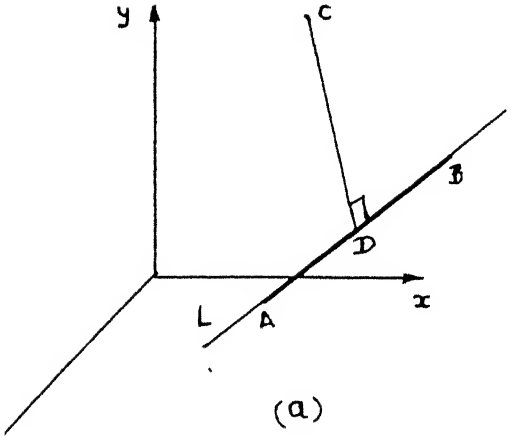
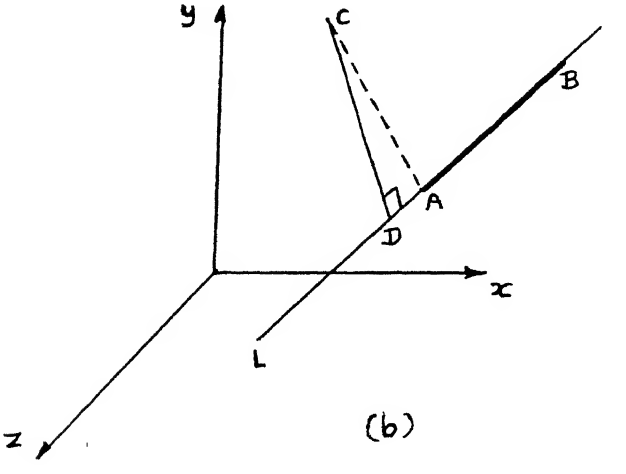


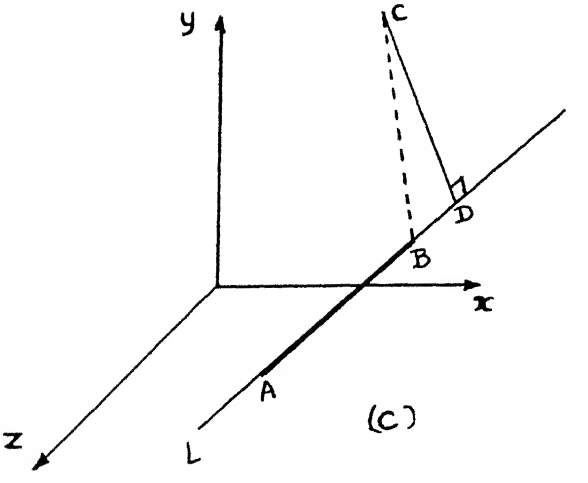
Figure A-3 Shortest distance between two skew lines.



(a)



(b)



(c)

Figure A-4 Shortest distance between a point and a line segment.

In such a case this becomes a problem of determining minimum distance between two skew segments in general For this two procedures are required, finding shortest distance between two lines and finding shortest distance of a point from a line

* Shortest Distance Between Two Lines - Let L_1 and L_2 be the two lines (figure A-3) Let points A and B be on L_1 and points C and D be on L_2 Let EF be the shortest distance segment. Let l_1,m_1,n_1 and l_2,m_2,n_2 be the direction cosines of L_1 and L_2 respectively which can be determined from the coordinates of points A,B,C,D

Now a general point on line L_1 will be given by

$$(x_A + l_1r_1 , y_A + m_1r_1 , z_A + n_1r_1) \quad \text{--- (A 7)}$$

and that on L_2 will be given by

$$(x_C + l_2r_2 , y_C + m_2r_2 , z_C + n_2r_2) \quad \text{--- (A.8)}$$

The direction cosines of a general segment connecting lines L_1 and L_2 will then be

$$(x_A - x_C + l_1r_1 - l_2r_2 , y_A - y_C + m_1r_1 - m_2r_2 , z_A - z_C + n_1r_1 - n_2r_2) \quad \text{--- (A 9)}$$

For this segment to be the shortest distance segment EF it has to be perpendicular to both L_1 and L_2 This gives

$$l_1(x_A - x_C + l_1r_1 - l_2r_2) + m_1(y_A - y_C + m_1r_1 - m_2r_2) + n_1(z_A - z_C + n_1r_1 - n_2r_2) = 0$$

--- (A.10)

and

$$l_2(x_A - x_C + l_1r_1 - l_2r_2) + m_2(y_A - y_C + m_1r_1 - m_2r_2) + n_2(z_A - z_C + n_1r_1 - n_2r_2) = 0$$

--- (A.11)

or $k_0 + k_1r_1 + k_2r_2 = 0$

and $k_3 + k_4r_1 + k_5r_2 = 0$

where

$$k_0 = l_1 (x_A - x_C) + m_1(y_A - y_C) + n_1 (z_A - z_C)$$

$$k_1 = l_1^2 + m_1^2 + n_1^2$$

$$k_2 = - l_1 l_2 - m_1 m_2 - n_1 n_2$$

$$k_3 = l_2 (x_A - x_C) + m_2(y_A - y_C) + n_2 (z_A - z_C)$$

$$k_4 = l_1 l_2 + m_1 m_2 + n_1 n_2$$

$$k_5 = - l_2^2 - m_2^2 - n_2^2$$

Solving , we get $r_1 = \dfrac{k_2 k_3 - k_0 k_5}{k_1 k_5 - k_2 k_4}$ and $r_2 = - \dfrac{k_4 r_1 + k_3}{k_5}$ --- (A 12)

k_0 through k_5 can be obtained from the coordinates of the points A,B,C and D and r_1 and r_2 can be evaluated. Using r_1 and r_2 in equations (A 7) and (A.8) we can get the coordinates of points E and F and hence the shortest distance can be determined

* Shortest distance between a point C and a line segment AB .- Let L be the line containing AB. Let CD be the perpendicular from C to line L Coordinates of a general point D on line L are $(x_A + lr , y_A + mr , z_A + nr)$ where l, m , n are the direction cosines of line L . The direction cosines of CD are then

$$(x_C - x_A - lr , y_C - y_A - mr , z_C - z_A - nr) \quad \text{--- (A.13)}$$

For this segment CD to be the shortest distance segment

$$l (x_C - x_A - lr) + m (y_C - y_A - mr) + n (z_C - z_A - nr) = 0 \quad \text{--- (A.14)}$$

From this equation r can be found out as

$$r = l (x_C - x_A) + m (y_C - y_A) + n (z_C - z_A) \quad \text{--- (A.15)}$$

and then putting r in equation (A.13) coordinates of point D can be found out

If D lies between A and B then the required shortest distance is length(CD) (see figure A-4 (a)).

If D lies outside AB on the A side then it is length(CA) (see figure A-4 (b))

and If D lies on the B side of AB then it is length(CB) (see figure A-4 (c))

Using the above two procedures, minimum axial distance available (SHORTDIST) between two line segments can be determined by a systematic procedure as below

Let L_1 and L_2 be the lines containing the segments AB and CD respectively Shortest distance between L_1 and L_2 (SDIST) is determined

If SDIST > MINDIST then the components do not interfere and no further investigation is necessary

If SDIST < MINDIST then coordinates of the points E and F are determined (figure A-5)

Following three possibilities arise

(i) Points E and F are on the line segments CD and AB respectively. In such a case SHORTDIST = SDIST and the two components definitely interfere (figure A-5 (a))

(ii) One of the two points E and F is outside and the other is inside the corresponding segments (figure A-5 (b)) Let F be outside AB and E be on CD. In this case it is determined on which side of segment AB is point F. If F is on the B side then shortest distance between point B and segment CD is SHORTDIST. If F is on the A side then the shortest distance between A and segment CD is SHORTDIST

If F is on segment AB and E is outside CD, similar procedure is followed using point E and segment AB.

(iii) Both the points E and F are outside the corresponding segments (figure A-5 (c)) In such a case it is determined on which sides of the corresponding

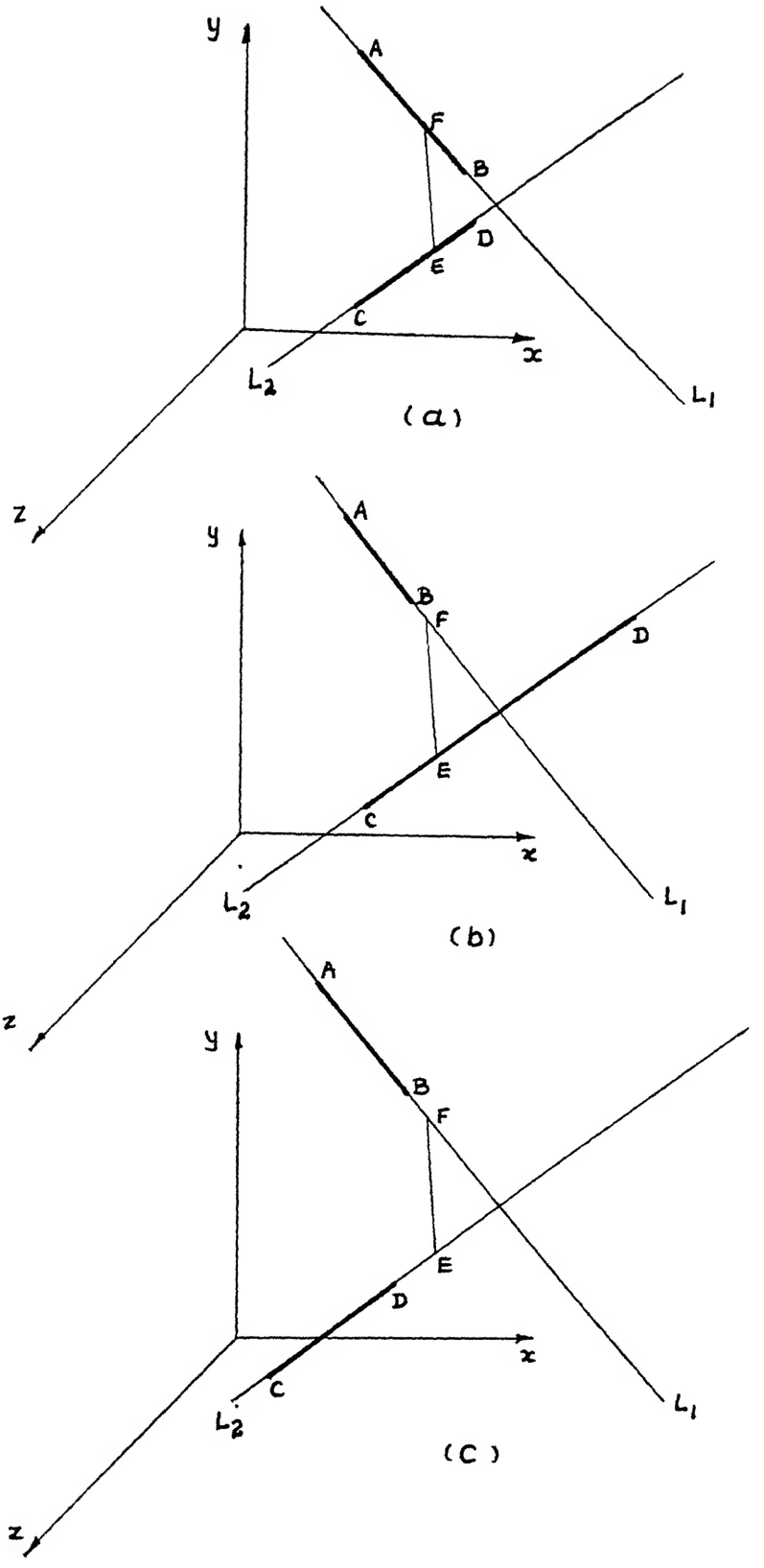


Figure A-5 Shortest distance between two skew line segments.

segments are the points E and F Let E be on the D side and F be on the B side then shortest distances from point D to AB and from point B to CD are determined SHORTDIST is the minimum of these two

A.2 Shortest Axial Distance Available Between a Straight Pipe Piece and a Pipe Bend (figure A-6)

Following procedure is adopted for this

* Shortest distance (SDIST) between centre of arc (O) and line segment CD is determined. If SDIST > r + MINDIST then the components do not interfere, and no furthur investigation is necessary

* A set of equidistant points on the arc AB are generated If diameter of the pipe is less than 150 mm, 6 points are generated, else 10 points are generated Shortest distance SDIST from each of these points to the line segment CD are determined If any of these distances is less than MINDIST, the components interfere; else they do not. This is an approximate approach, but if some additional clearance is used while calculating MINDIST, then the procedure works very well To reduce the inaccuracy more number of points are generated for larger diameter pipe bends

A.3 Shortest Axial Distance Available Between Two Pipe Bends (figure A-7)

If length (O_1O_2) > r_1 + r_2 + MINDIST then the components do not interfere, and no furthur investigation is necessary. Otherwise, two cases as below have to be checked separately

(i) The axis arcs of the bends are on parallel planes .- For this a procedure to determine the shortest distance from a point to a coplaner 90° arc of a circle

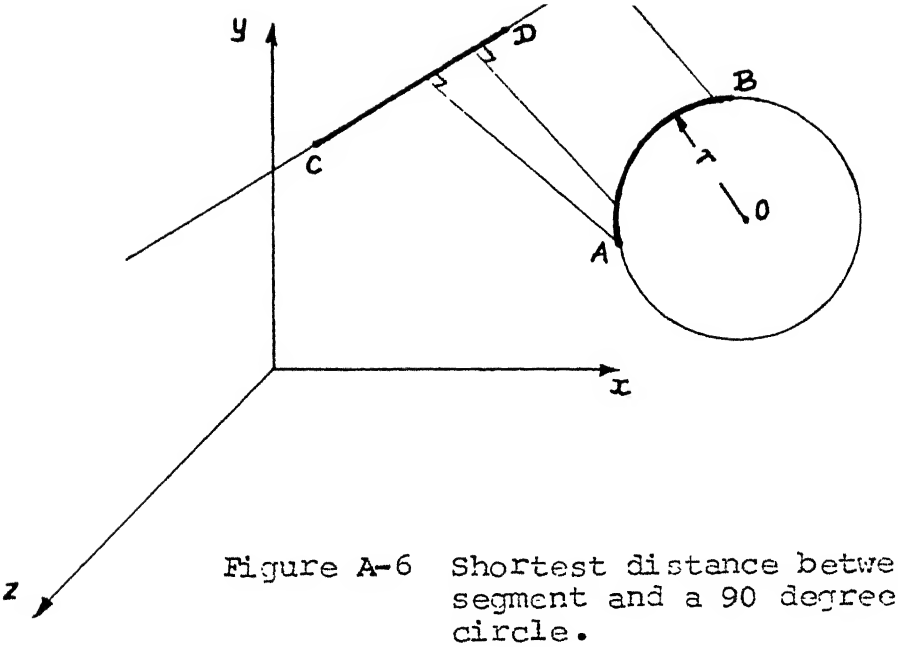


Figure A-6 Shortest distance between a line segment and a 90 degree arc of a circle.

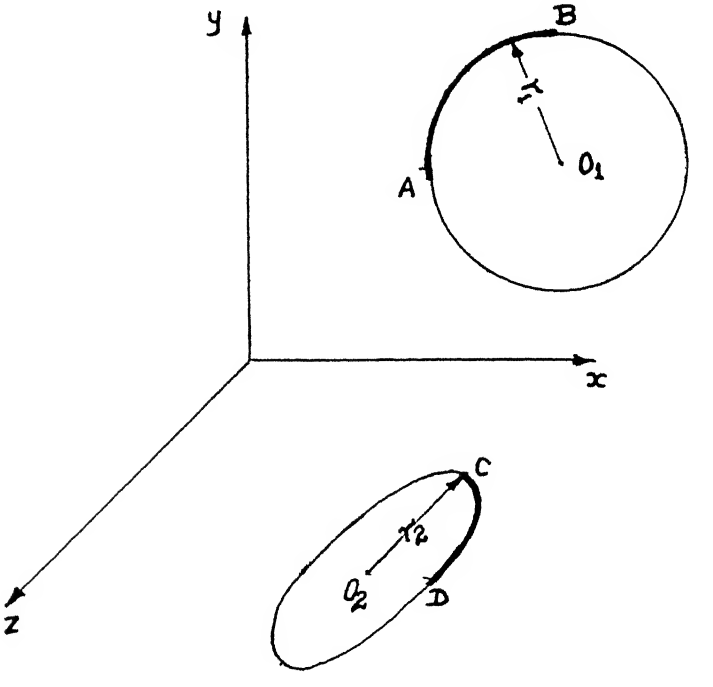


Figure A-7 Shortest distance between two 90 degree arcs of circle.

is required, which is as follows :

Consider an arc AB and a point C as shown in figure A-8 The points E and F can be easily determined At the most one of the two points can lie on the arc AB Whether a point E lies on the arc AB or not can be determined by comparing the three distances AB, AE and BE Two possibilities arise

* If one of the points, say E is on arc AB (figure A-8 (c), (d)) then

SHORTDIST = min (length(CB), length(CA), length(CE))

* If both the points are outside arc AB (figure A-8 (a), (b)) then

SHORTDIST = min (length(CB), length(CA))

Using the above procedure the minimum distance between two 90° arcs of circle on parallel planes can be determined as follows

The two arcs are assumed to be on the same plane A set of points (number depending on the diameter of the pipe) on one of the arcs are generated. Shortest distance of each point from the other arc is obtained as explained above. These distances are then updated by including the normal distance between the two planes If any of these distances is less than MINDIST then the components interfere; else they do not

(ii) The axis arcs are on mutually perpendicular planes - In this case two sets of points on the respective arcs are generated Again the number of points for each arc depends on the diameter of the respective pipe. SHORTDIST is then the minimum of the distances between all combinations of these points.

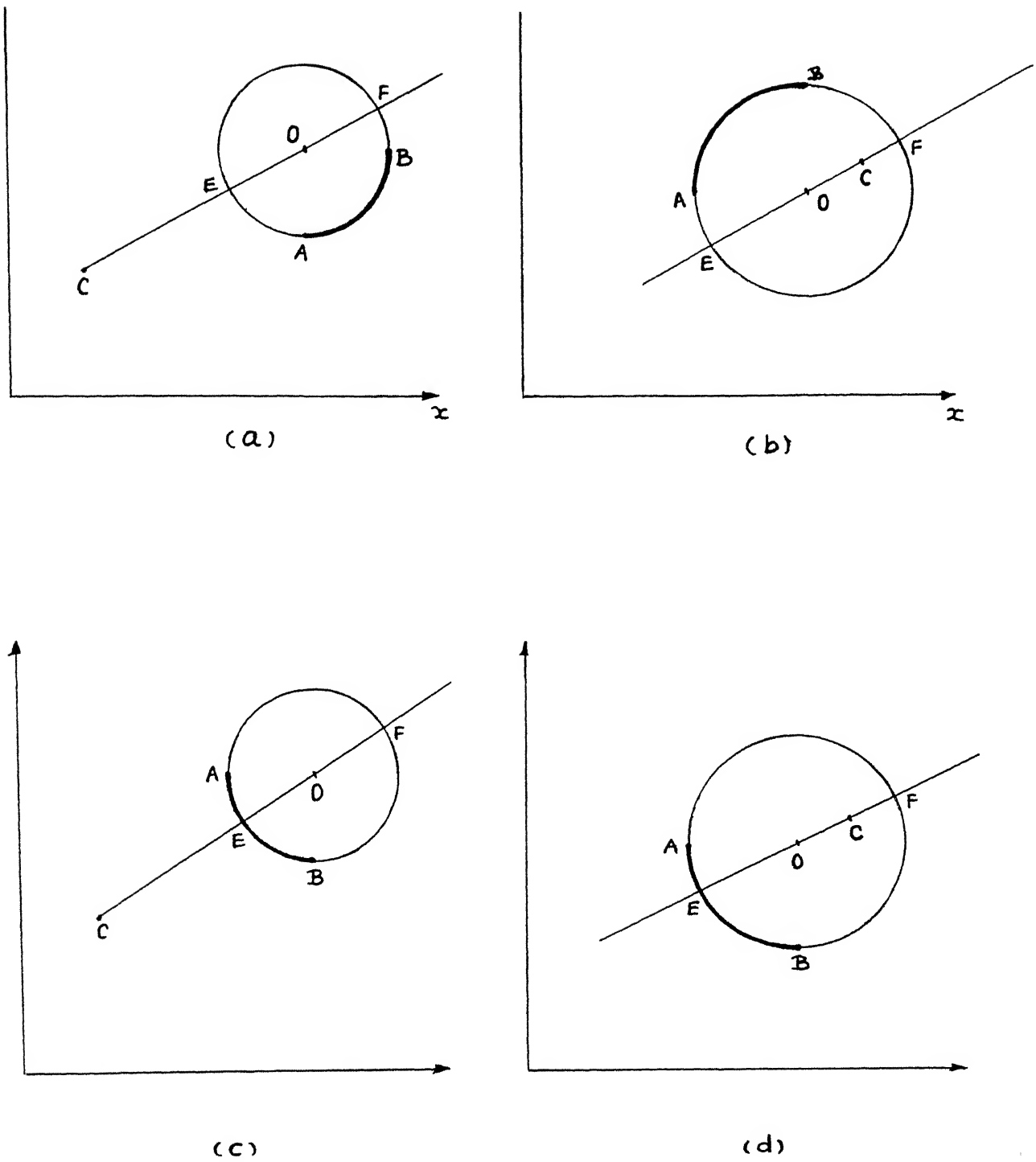
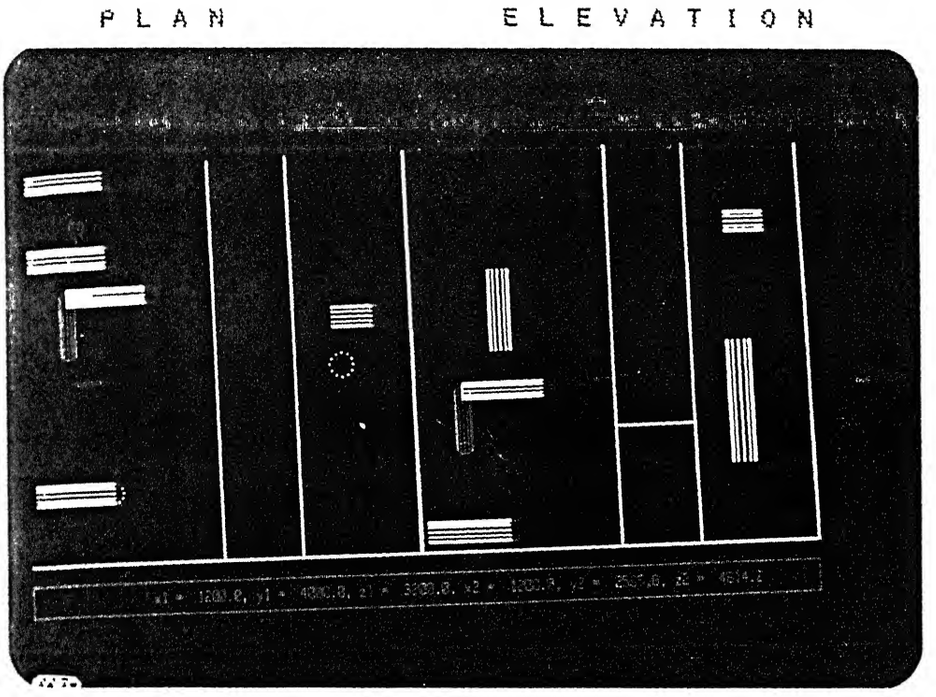
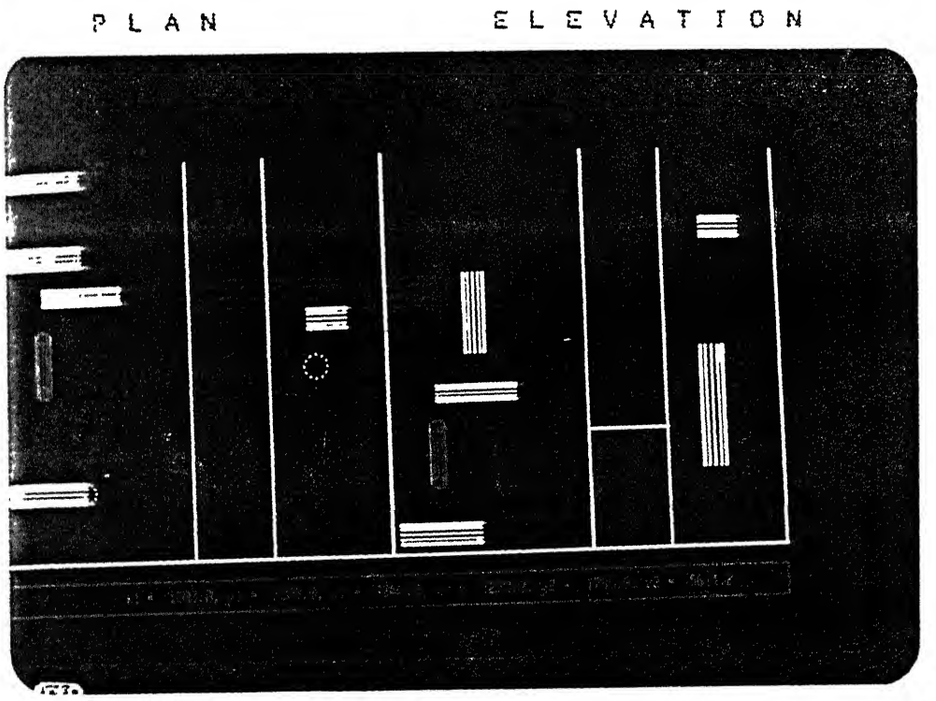


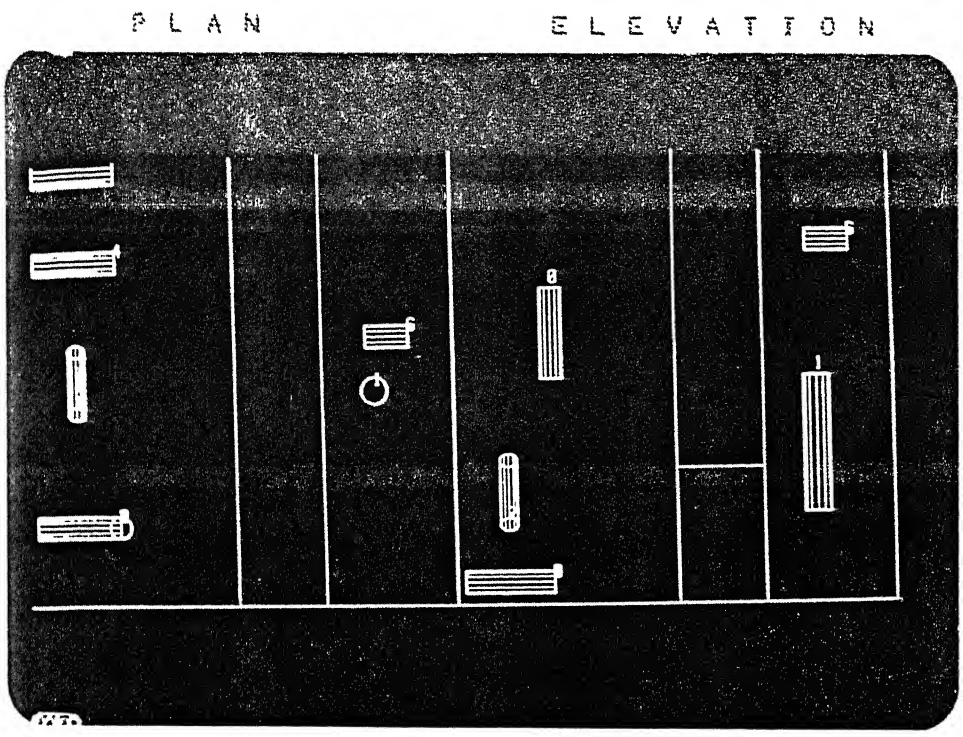
Figure A-8 Shortest distance between a 90 degree arc of a circle and a coplaner point.



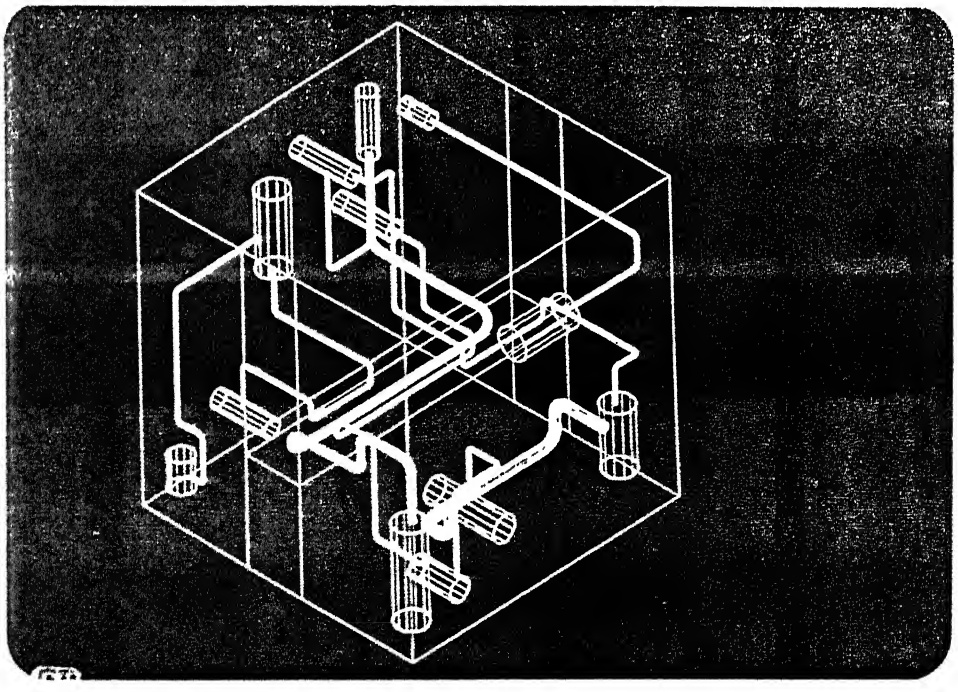
Picture 5.1 : Movement of vessel (Rotation)



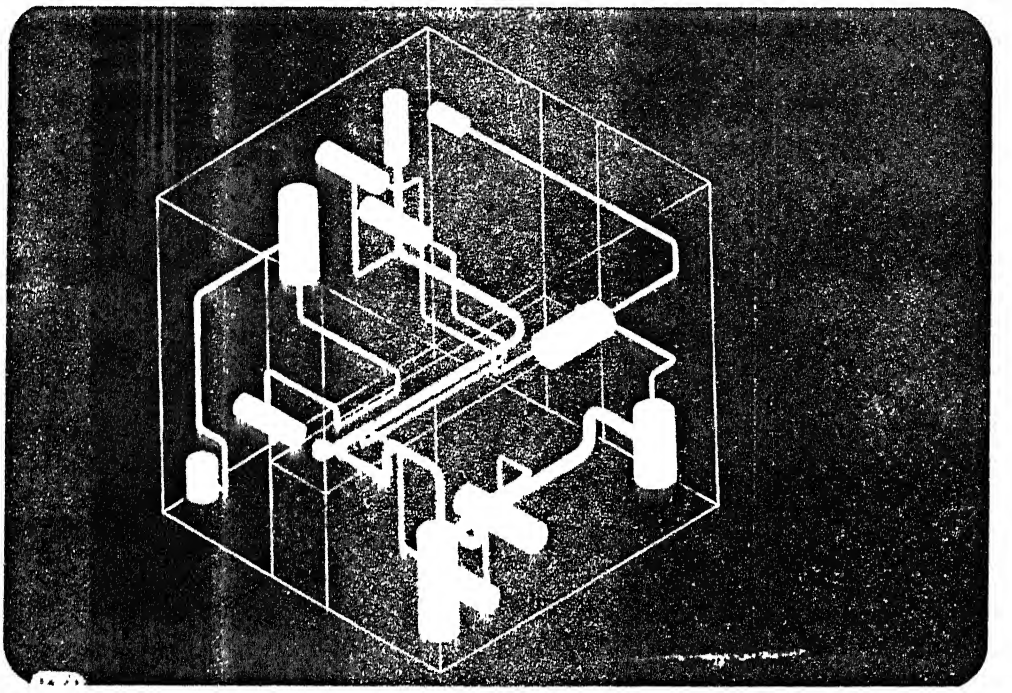
Picture 5.2 : Movement of vessel (Translation)



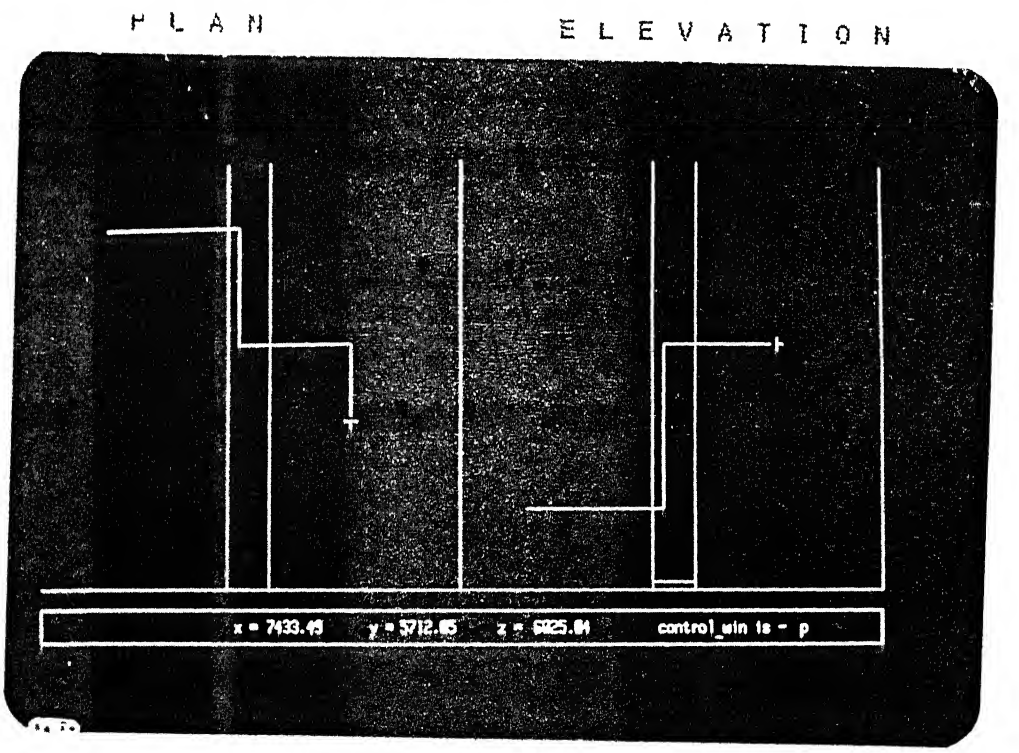
Picture 5.3 : Movement of vessel (New Position)



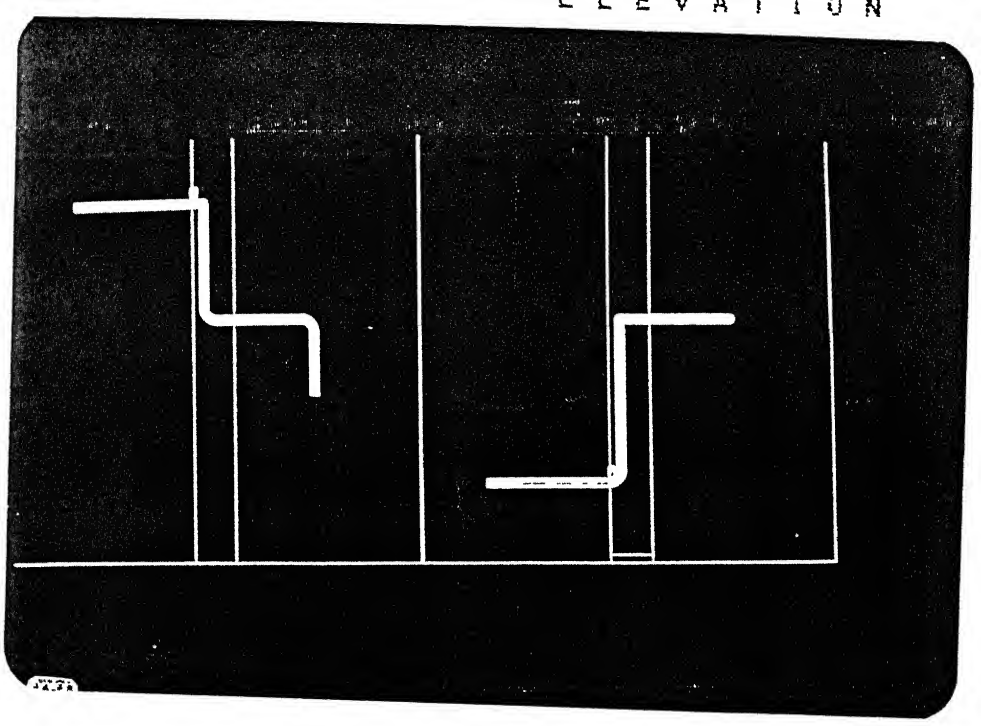
Picture 5.4 : Display. Isometric View (Wireframe)



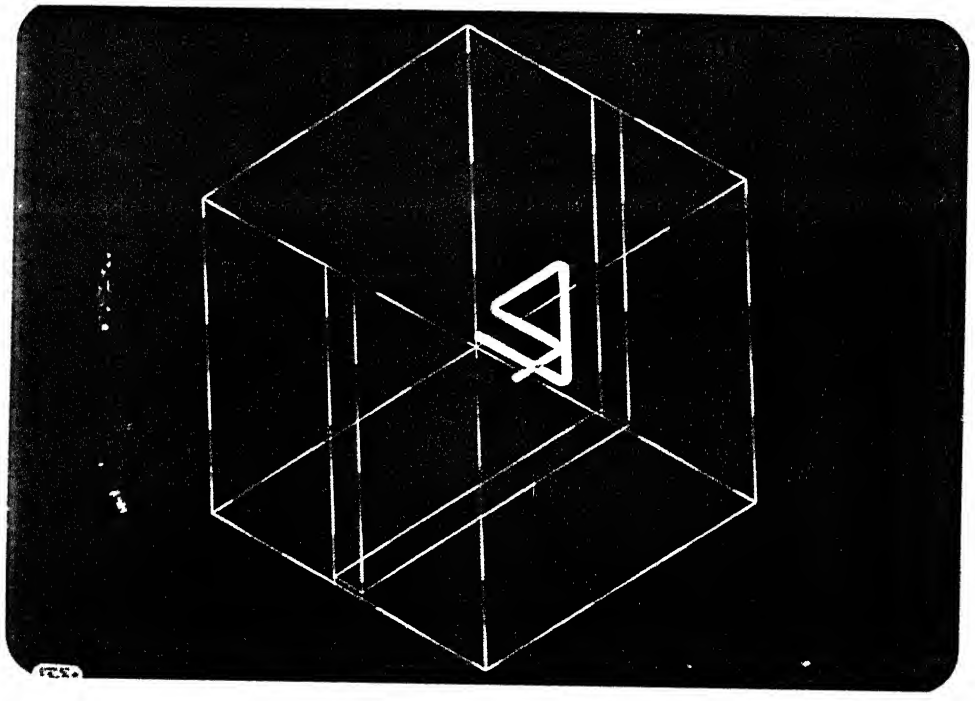
Picture 5.5 : Display. Isometric View (Surface Model with hidden surfaces removed).



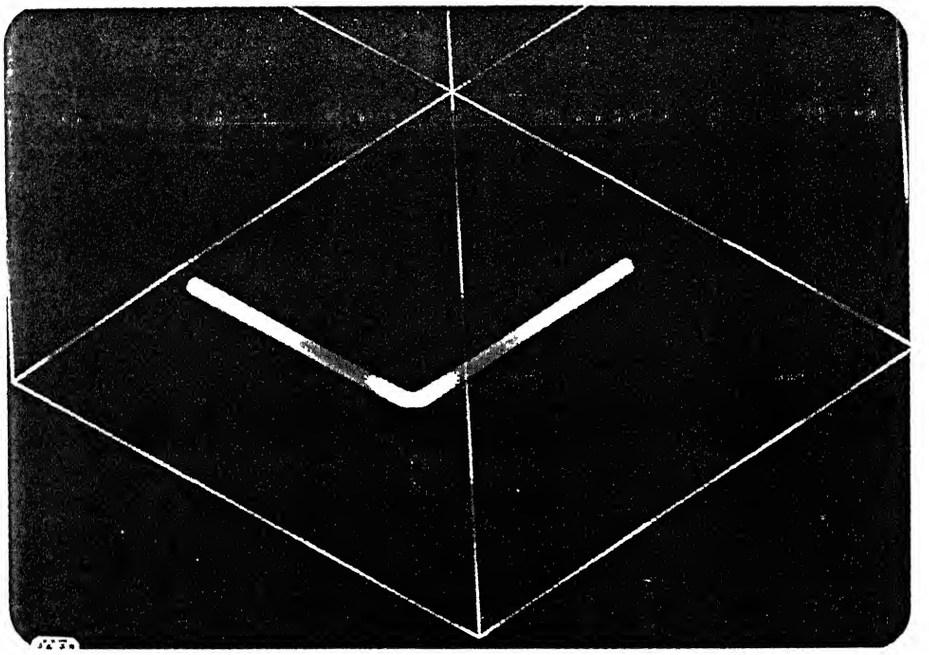
Picture 5.6 : Manual Routing (Sequence of points entered by the user).



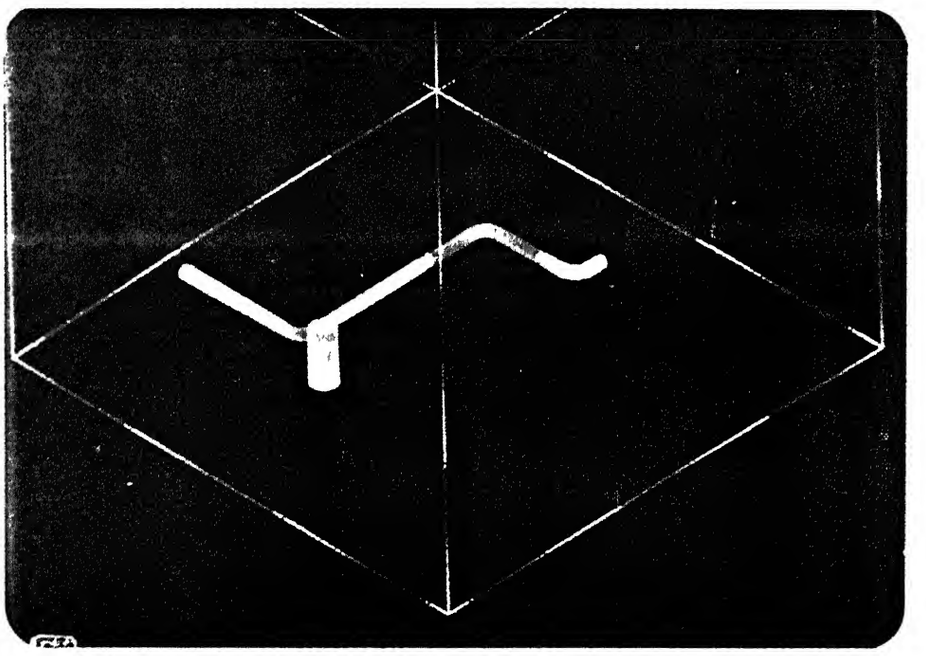
Picture 5.7 : Manual Routing (Two views of the routed pipe)



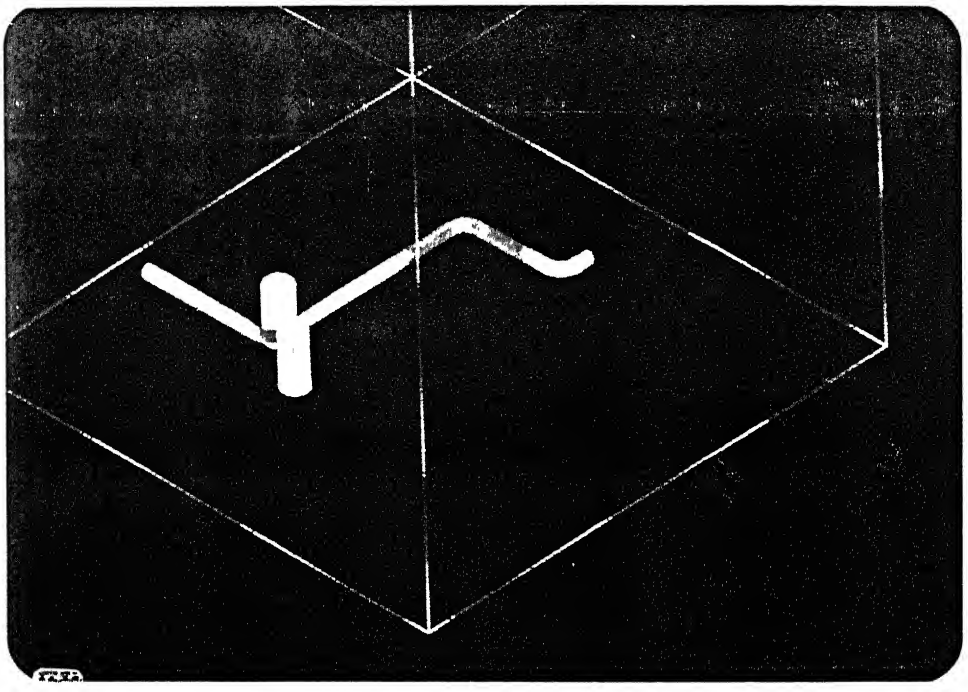
Picture 5.8 : Manual Routing (Isometric view of the same pipe).



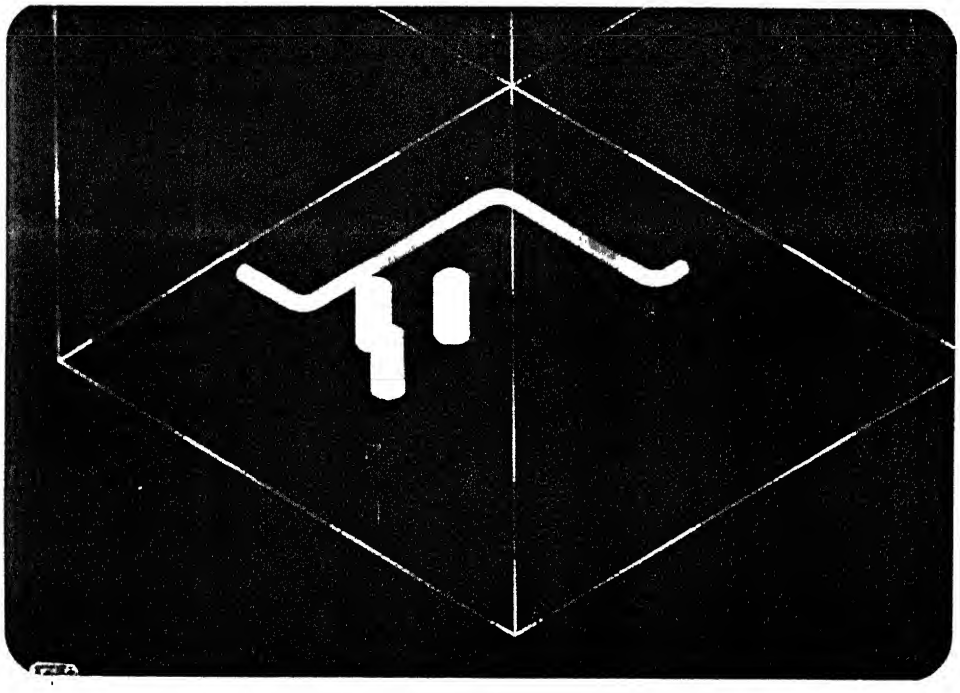
Picture 5.9 : Autorouting between two points (number of obstacles = 0).



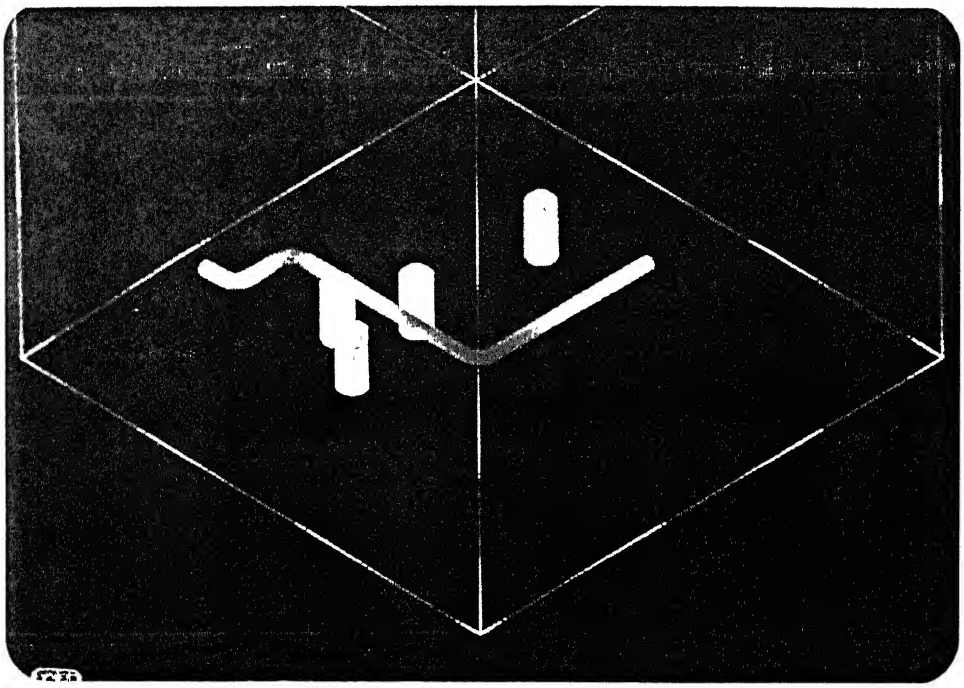
Picture 5.10 : Autorouting between two points (number of obstacles = 1).



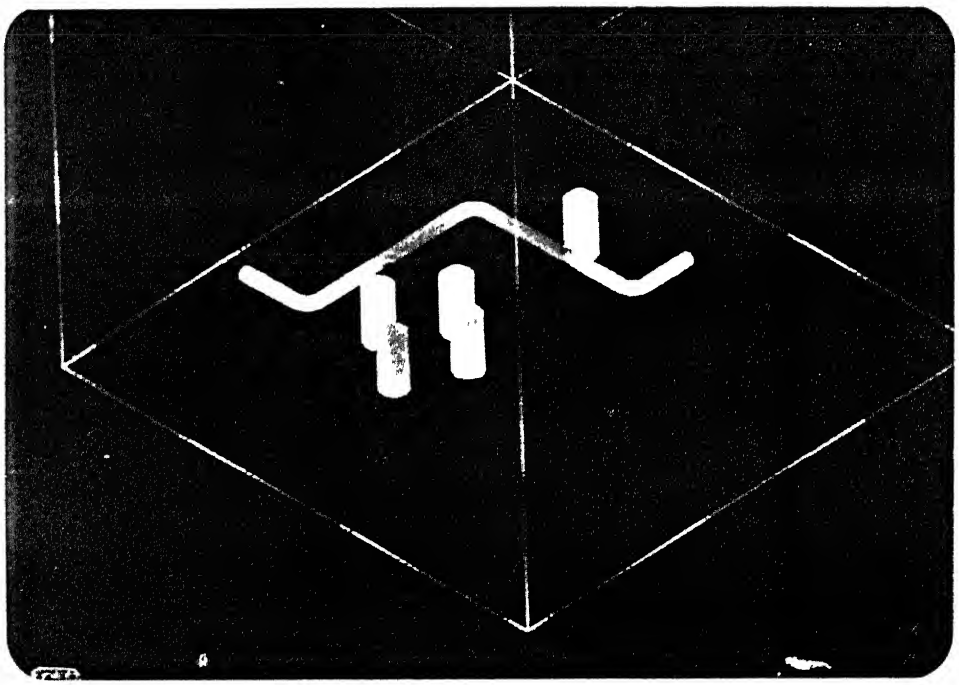
Picture 5.11 : Autorouting between two points (number of obstacles = 2).



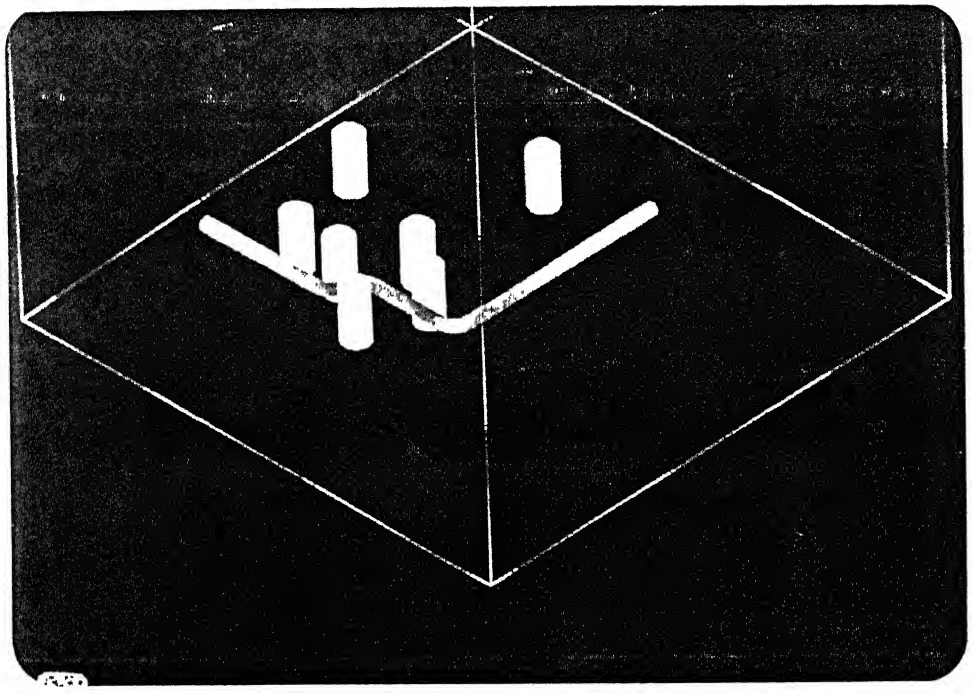
Picture 5.12 : Autorouting between two points (number of obstacles = 3).



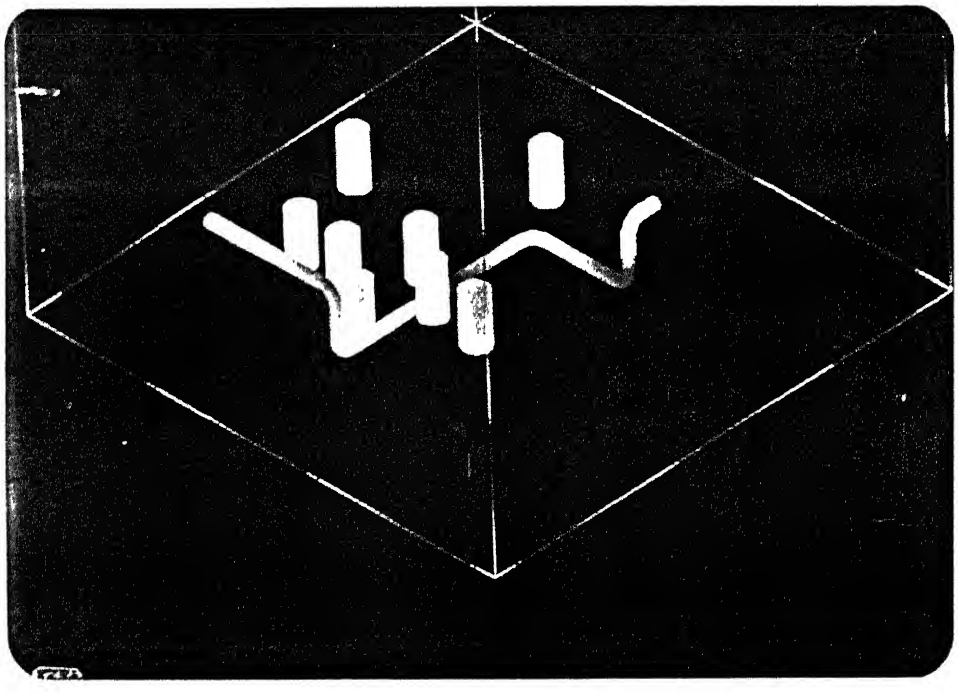
Picture 5.13 : Autorouting between two points (number of obstacles = 4).



Picture 5.14 : Autorouting between two points (number of obstacles = 5).



Picture 5.15 : Autorouting between two points (number of obstacles = 7).



Picture 5.16 : Autorouting between two points (off-plane routing. Number of obstacles = 8).